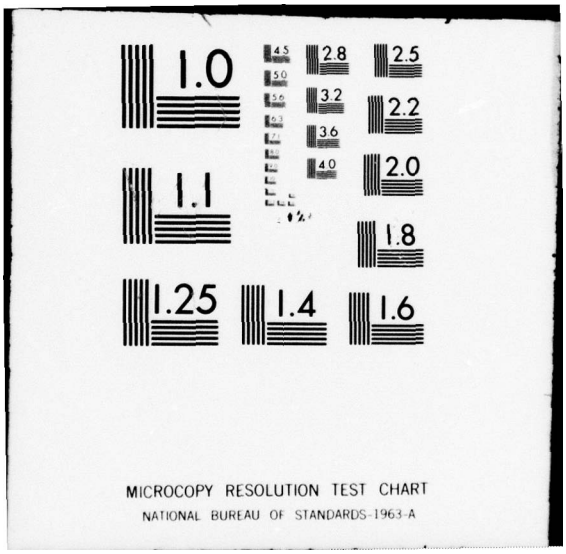
AD-A058 629 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART I. (DESCRI--ETC(U)
JUL 78 F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-130-VOL-1 NL

1 OF 3
AD
A058629


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ESD-TR-78-130, Vol. I

LEVEL

10

USER REQUIREMENTS LANGUAGE (URL)
USER'S MANUAL PART I (DESCRIPTION)
H6180/MULTICS/VERSION 3.3

ISDOS Project
University of Michigan
Department of Industrial and Operations Engineering
Ann Arbor, Michigan 48109

July 1978

Approved for Public Release;
Distribution Unlimited.

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

This Technical Report has been reviewed and is approved for publication.

Ann Melanson

ANN MELANSON
Project Engineer

C. J. Greene Jr.

CHARLES J. GREENE, Jr., Lt Col, USAF
Chief, Technology Applications Division

FOR THE COMMANDER

Eric B. Nelson

ERIC B. NELSON, Colonel, USAF
Acting Director, Computer Systems Engineering
Deputy for Technical Operations

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-78-130, Vol. I VOL-1	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER: Technical rept.	
4. TITLE (and Subtitle): USER REQUIREMENTS LANGUAGE (URL) User's Manual, Part I, (Description) H6180/Multics/Version 3.3.	5. TYPE OF REPORT & PERIOD COVERED: July 1976 to March 1977. Manual
	6. PERFORMING ORG. REPORT NUMBER: CDRL Item 021
7. AUTHOR(s): ISDOS Project	8. CONTRACT OR GRANT NUMBER(s): F19628-76-C-0197
9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of Michigan, Department of Industrial & Operations Engineering, Ann Arbor, MI 48109	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: PE64740F, Project 2237
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Technical Operations, Electronic Systems Division, Hanscom AFB, MA 01731	12. REPORT DATE: July 1978
	13. NUMBER OF PAGES: 212
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 224p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

DDC

SEP 14 1978

F

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer-Aided Design
Information Processing
Information System Requirements
Requirements Analysis
Requirements Language
Requirements Specification
Specification Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is part of a series that deals with a Computer-Aided Design and Specification Analysis Tool (CADSAT). The purpose of the tool is to describe the requirements for information processing systems and to record such descriptions in machine-processable form. The major components of CADSAT are the User Requirements Language (URL) and the User Requirements Analyzer (URA) which can operate in an interactive computer environment. This report, Part I and Part II, describes how the formal URL may be used to define systems. It explains the language statements available, their use and application on a Honeywell 6180 Multics Computer.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

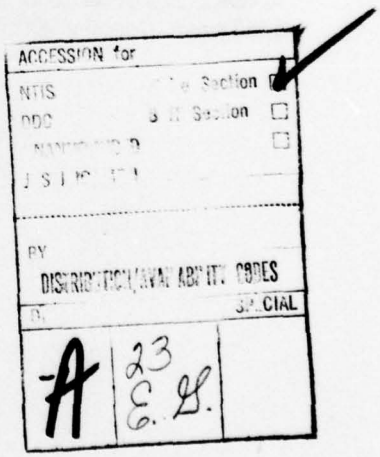
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403 871

LB

PREFACE

This manual describes the User Requirements Language (URL) to be used with Version 3.3 of the User Requirements Analyzer (URA). The manual consists of two volumes which are referred to as Part I and Part II in the documentation. Part I gives a detailed description of the URL statements available and their use. Part II is a reference manual which gives the proper syntax for each statement.



User Requirements Language (URL) is a language for describing an Information Processing System (IPS). A Problem Statement (PS) in URL can be used to describe the "present" system or to state requirements that a "proposed" target system is to fulfill. Describing the "present" system is helpful in finding where redundant information exists, standardizing procedures, etc., and also forms the basis for designing "proposed" systems. In describing a "proposed" system, the Problem Statement can be considered as the specifications for the succeeding stages in the system life cycle, i.e., in the physical design and construction phases.

Requirements for proposed information processing systems are usually described in the Logical System Design phase sometimes called the "feasibility study." The end result of the logical system design process is a description of a proposed system and a benefit/cost analysis of the value of building it. The process itself may be accomplished in many different ways but the particular method chosen does not affect the form of the final result. What constitutes a satisfactory description of the proposed system is not affected by whether the process is carried out manually or with computer aids (except for the fact that the computer-aided method can result in the description itself being stored in a computer-aided processable form).

The purpose of the manual is to describe how URL may be used to describe systems. It may be used as an introduction to the use of URL and is also used as a text in URL courses. It contains the complete syntax and semantics of URL as well as providing guidelines on how these are intended to be used. A more precise statement of URL is given in the User Requirements Language, Language Reference Manual, Part II. Additional information in the use of URA is given in the User Requirements Analyzer User's Manual.

PART I USER REQUIREMENTS LANGUAGE MANUAL

PART I USER REQUIREMENTS LANGUAGE MANUAL

PART I USER REQUIREMENTS LANGUAGE MANUAL

1. INFORMATION PROCESSING SYSTEM DESCRIPTION

Information Processing Systems of all types exist in organizations today. They serve to store, retrieve, manipulate or organize information in some manner to meet a particular organization's needs. For this reason, the design and operation of one of these systems are particular to a given organization. To conform with the changing environment, an organization must develop new systems, modify existing systems and terminate obsolete ones. This can require a major effort of the organization to design systems and maintain documentation of a system once it is operational.

1.1 Introduction

1.1.1 System Life Cycle

An information processing system has a life cycle which begins with the initial conception of need of the system, proceeds through determination of requirements for the system, (logical system design), physical system design, detailed design and construction, operation, modification and maintenance and finally, termination of system operation.

1.1.2 Documentation

At each step of the life cycle, some form of documentation is needed by the organization. The documentation consists of a complete and comprehensive description of the proposeed (or target) system. In addition, the organization of which the system is part must be at least partially described; and the project defining and designing the system must also be described.

1.1.2.1 What has to be Documented

No matter what type of system is to be designed or who is designing it, there exist some features or components which are common to all systems and that must, therefore, be included in its documentation. Together, these common characteristics can be regarded as constituting a model of the system. This model is shown in Figure 1.

The basic purpose for constructing a system is to serve some organization. Usually, a new system is required to solve some "problem" within the existing system.

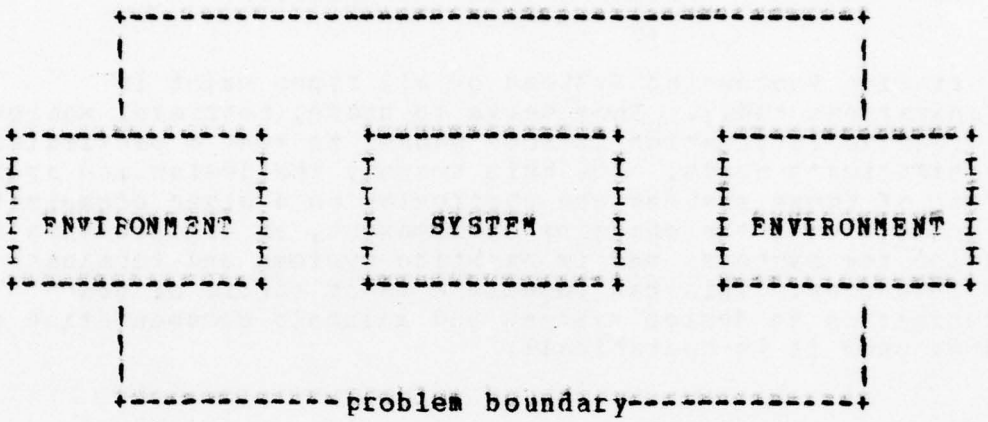


Figure 1: The Problem

The task of the system builders is to accurately define the problem so that a solution may be implemented. The problem, therefore, has three basic components or elements:

- An environment in which the problem occurs. Those parts of the organization which directly interface with the problem must be included in the description.

- The target system which is being described to resolve the problem. The word "target" connotes a "proposed" rather than an existing system. The relation between the environment and the target system is shown in Figure 1.

- The Project assigned the task of defining the problem, adequately documenting the requirements, designing, constructing and installing the system.

All of the elements must be documented in sufficient detail to meet the needs of the organization. To accomplish this, the elements must be broken down to smaller components. These in turn must be broken down or subdivided into smaller elements. The elements at all levels are called the "system description elements."

1.1.2.2 Purposes of Documentation

The description of the problem throughout the life cycle is usually referred to as "documentation." Such documentation must serve a number of different purposes:

- The system builders must have a record of what they have done.

- The organizations within the environment of which the system is to serve must have a description to assure themselves that

the prepared system will satisfy their needs.

- The management of the organization that is providing the resources must know what they are approving.

- The system builders who will continue the development, construction, operation and maintenance of the system must all have documentation from which to carry out their tasks.

1.1.2.3 Forms of Documentation

To serve their needs, most organizations have developed standard documentation procedures consisting of very general to very specific guidelines in producing documentation. Some organizations use commercial documentation packages or documentation techniques in hopes of producing more complete, correct and consistent documentation.

It is standard practice to record the description of the system in formal documents corresponding to various stages of the life cycle known by such names as the system definition report, system requirements report, system design report, programming documentation, user's manual, etc.

These documents are normally in narrative form, supplemented by diagrams, flow charts, lists, glossaries, cross references, etc.

1.1.2.4 Characteristics of System Documentation

Information Processing Systems are large and complex and regardless of who produces the documentation or what graphical aids, such as flow charts are used, it will have several features that make the process of documentation different.

- Size. Complete documentation of a system may consist of many thousand of pages of charts, tables, code listings, user guides, project plans, etc.

- Complexity. Any piece of information about some aspect of the system or the project may be related to many other pieces.

- Multiple users with different needs. Each of the users of the documentation, as noted above, need the documentation of some aspects of the system at different levels of detail.

- Changeability. The documentation must be constantly updated as changes occur in the organization or in the system. Any change, because of the complexity, can affect the documentation in many places.

1.1.3 Process of Documentation

1.1.3.1 Manual

Someone must be responsible for this documentation. It is often the task of the analyst to do this. In other cases, it may be a technical writer who must obtain the information from other sources (analysts, management, memos, etc.) in order to produce the documentation of the system. The technical writer has the disadvantage of not being directly involved in the system development effort. The analyst has the advantage of being directly involved with the system yet is sometimes too close to it to present a complete description. The documentation is usually produced manually regardless of who is doing it.

1.1.3.2 Computer-Aided Documentation - URL/URA

A computer-aided approach to system documentation can be an improvement over the manual methods by using the power of the computer to store large quantities of data and to manipulate complex relationships.

To take advantage of the potential benefits, a computer-aided documentation system should have the following characteristics:

a) A formal language flexible enough to describe any type of information processing system.

b) A translator which takes the formal language statements as input and stores it in some processable form in the computer (i.e., on disk or tape).

c) A data base in which the information interpreted from the language statements is stored.

d) A report generator which allows information in the data base to be retrieved, analyzed and formatted as reports.

e) An update facility which allows information in the data base to be added to, modified, or deleted. Before any information in the data base is updated, checks must be made for consistency and correctness so that accuracy of the information in the data base is maintained.

The advantages of using such a computer-aided technique versus a manual method are:

a) Though information is interrelated with other information, there is only one occurrence of each piece of information in the data base. If this piece of information is modified, the contents of the data base are modified to reflect the change.

b) The Language has a finite number of statements which may be

specified and syntax and semantic rules for each of these statements. This allows persons documenting systems to give precise descriptions which are much less subject to misinterpretation.

c) Once the data base has been modified, all reports generated using it are up-to-date.

d) The reports generated are designed to view the system (as described in the data base) at various angles. One particular report may present high level structural information, another may present the manner in which low level data is manipulated in the system, and still others may present lists of names, dictionaries, etc.

e) Some reports may present results of complex analysis based on the contents of the data base. Analysis may consist of checks for completeness or consistency in the system description at any point in time.

1.1.3.3 URL/URA

URL is a computer-processable language designed primarily to describe a target system during its formative stage (i.e., during the determination of requirements phase in the system life cycle). It also contains facilities for describing those parts of the organization interfacing with the system and those parts of the project which are relevant to the description of the target system. The URL description of a system consists of a combination of formal statements (allowed by the language) supplemented by narrative descriptions.

The User Requirements Analyzer (URA) is a software package which processes the URL statements and acts as an interface between the problem definers and the information stored as the URL description.

Organizations usually require that the documentation of a proposed system include a "system requirement report." This document contains a detailed description of the target system, information about the manner in which the system interfaces with the organization, and some description of the project designing the system including estimates of costs, resources required and completion time, etc. URL is designed to state the type of information which appears in the system requirement report and when a problem is completely described, essentially all the information for the system requirement report is contained in the URA data base.

1.1.4 Introduction to URL - A Formal System Description Language

The description of a system involves describing "objects," the "properties" of these objects, and the "relationships" among the objects.

In Section 1.1.2 these "objects" were referred to as "system description elements" representing some physical or conceptual thing in the target system. Examples of "objects" are "logical collections" of data, the "processes" which define how the data is derived, etc. Each "object" defined in the target system must be assigned a unique name and classified by the "type" of object which it is. URL, for example, allows approximately 30 different types of "objects."

"Properties" of an "object" consist of statements describing that "object."

"Relationships," on the other hand, describe connections among "objects." To say that object A uses objects B and C specifies "relationships" among these objects. There are approximately 75 different "relationships" that may be used in URL.

An example of a description of a (very simple) system is given in Section 1.2. A full description of the types of "objects" that may be defined in URL, their purposes and usages, is given in Section 1.3. The "relationships" allowed in URL are given in Section 1.4 along with information on how these relationships relate to the overall system description and special considerations involved when using them.

1.2 Example

This section illustrates the fundamental concepts of objects, names of objects, types of objects and relationships among objects through the use of a simple example. The process of computer-aided documentation is also shown. (Properties of objects are not illustrated in this example.)

1.2.1 Narrative Description

The following is a typical narrative description of a particular system:

> "A system called payroll processing takes employee information which comes from departments and employees and produces outputs which go to the departments and employees. The system also maintains payroll master information."

The information in such a narrative description is usually shown graphically as in Figure 1.2.1 or in Figure 1.2.1.1.

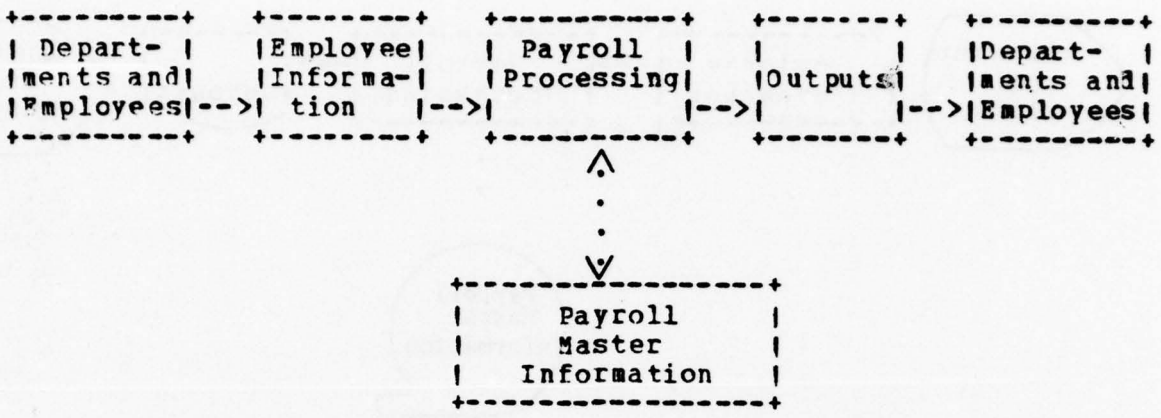


Figure 1.2.1
System Flowchart for PAYSYSTEM

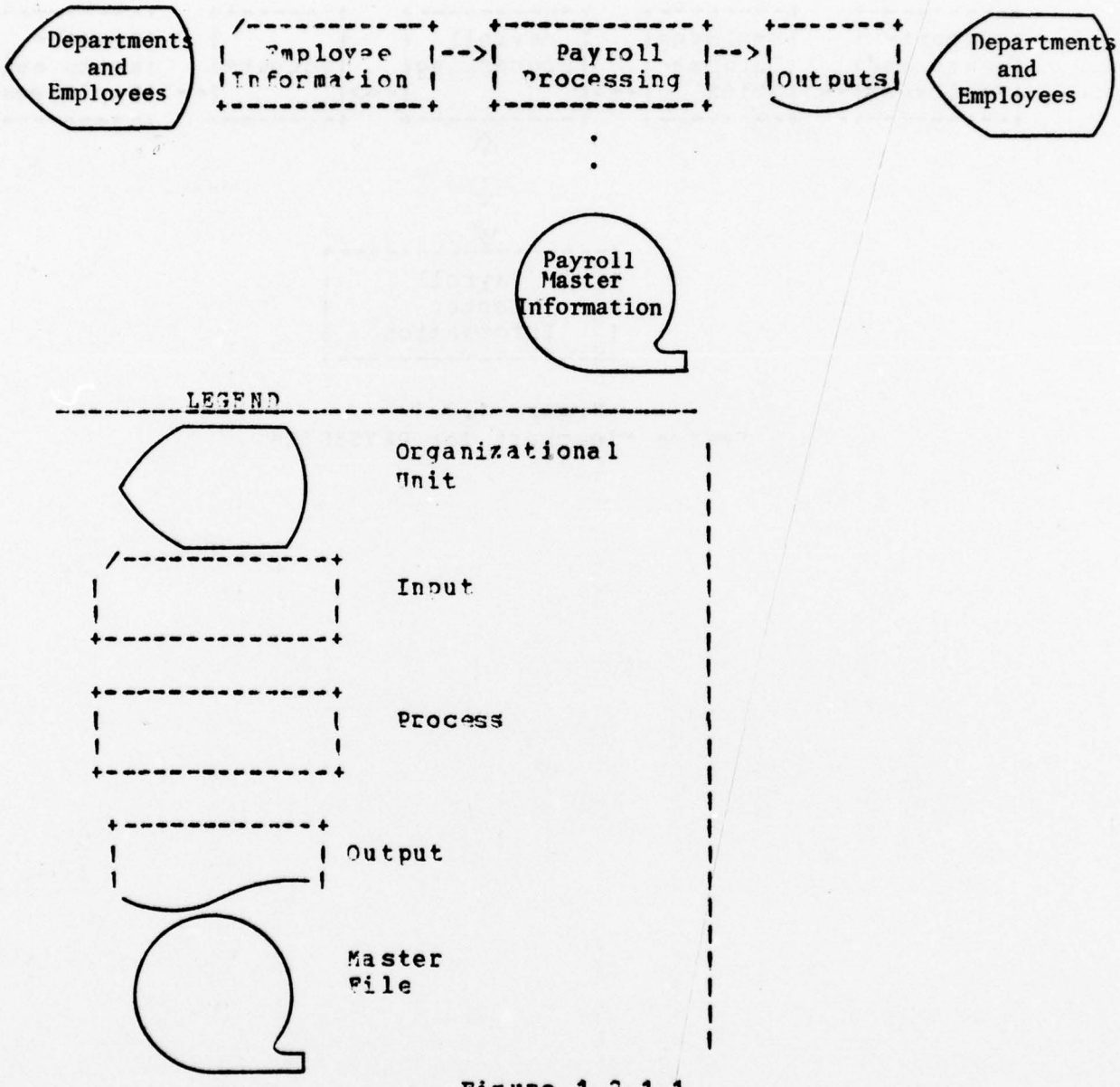


Figure 1.2.1.1
System Flowchart for PAYSYSTEM
Using Standard Flowchart Symbols

1.2.2 Identification of Objects

The first step in using URL is to identify the objects in the system being described. This can be done for the above example by underlining them in the narrative description:

> "A system called payroll processing takes employee information which comes from departments and employees and produces outputs which go to the departments and employees.

The system also maintains payroll master information."

1.2.3 Object Names and Types

Each of the defined objects has a unique name and each of these objects is described in a different context; "employee information" represents information passing from "departments and employees" to "payroll processing," "payroll master information" represents information manipulated by "payroll processing," etc. In effect, each of these objects represent different types or classes of objects. For example, in URL, the type of object corresponding to that suggested by "employee information" is an INPUT, "payroll master information" is a SET, etc. The following table relates each of the objects defined in the narrative description with a corresponding URL name and object type:

Narrative	URL Name	URL Object Type
payroll processing	payroll-processing	PROCESS
employee information	employee-information	INPUT
departments and employees	departments-and-employees	INTER-FACE
outputs	paysystem-outputs	OUTPUT
payroll master information	payroll-master-information	SET

URL does not allow blanks in the names of objects; dashes are normally used to connect names consisting of more than one word. In an effort to keep the names used as meaningful as possible, "qualified" names such as "paysystem-outputs" (instead of "outputs") are encouraged.

1.2.4 Identification of Relationships

The next step in using URL is to identify the relationships among the objects which have been identified. The relationships implied in the example narrative description are underlined:

> "A system called payroll processing takes employee information which comes from departments and employees and produces outputs which go to the departments and employees. The system also maintains payroll master information."

The following relationships have been identified:

Relationship	Between	and
takes	payroll processing	employee information
comes	employee information	departments and employees
produces	payroll processing	outputs
go	outputs	departments and employees
maintains	payroll processing	payroll master information

There are a finite number of relationships that may be described by URL. By taking into account the types of objects defined in the above example and the relationships that URL allows among those objects, the following correspondence between the narrative description relationships and the URL relationships can be made:

Narrative relationship	URL relationship
takes	RECEIVES
comes	GENERATED BY
produces	GENERATES
go	RECEIVED BY
maintains	UPDATES

The description of the system using URL terminology is:

Object	Relationship	Object
payroll-processing	RECEIVES	employee-information
employee-information	GENERATED BY	departments-and-employees
payroll-processing	GENERATES	paysystem-outputs
paysystem-outputs	RECEIVED BY	departments-and-employees
payroll-processing	UPDATES	payroll-master-information

1.2.5 URL Format

The object type of a particular named object can be explicitly defined by a URL statement. For the above example, the following URL statements may be used to define the object type of "payroll processing," "employee information" and "departments and employees."

```
PROCESS      payroll-processing;
INPUT        employee-information;
INTERFACE    departments-and-employees;
```

Since a particular object may be involved in several relationships the format for specifying relationships is made as simple as possible. For any object defined via a statement declaring its object type (as above) those relationships the object is involved in may be listed after this statement along with the corresponding objects in the relationship. The URL format to specify the relationships "payroll processing" is

involved in is:

```
PROCESS          payroll-processing;
  RECEIVES       employee-information;
  GENERATES      paysystem-outputs;
  UPDATES        payroll-master-information;
```

1.2.6 URA Outputs

One complete URL problem statement for the example is shown below. (There are many ways in which all of the information could be stated. They are all equivalent as far as URA is concerned.)

```
  INPUT          employee-information;
  OUTPUT         paysystem-outputs;
  SET            payroll-master-information;
  INTERFACE      departments-and-employees;
    GENERATES      employee-information;
    RECEIVES       paysystem-outputs;
  PROCESS        payroll-processing;
    UPDATES        payroll-master-information;
    RECEIVES       employee-information;
    GENERATES      paysystem-outputs;
```

Once these statements have been entered into the URA data base, URA can be used to generate a number of "standard" outputs. Figure 1.2.6 shows one of these outputs called the FORMATTED PROBLEM STATEMENT. This report contains all information stored about selected objects in the data base. In this instance, the report has been generated for all the objects defined in the data base.

The format of the information in the FORMATTED PROBLEM STATEMENT is the same as that specified when describing the example in URL. The report also presents all implied relationships as well as the explicitly defined ones. This is the reason that, though only five relationships were given in the example, ten are presented in the FORMATTED PROBLEM STATEMENT. To say that 'payroll-processing' RECEIVES 'employee-information' implies that 'employee-information' is RECEIVED BY 'payroll-processing,' etc. These are called complementary statements and when describing a system in URL, the choice of which of the two complementary relationships to be used is arbitrary. (The information stored in the data base is exactly the same.) The following are the complementary relationships used in the example:

Relationship	Complementary Relationship
RECEIVES	RECEIVED BY
GENERATES	GENERATED BY
UPDATES	UPDATED BY

Figure 1.2.6.1 presents an example of a graphical output that may be obtained from URA. This particular example shows the relationships 'payroll-processing' is involved in. All objects are represented by rectangles with the name of the object within the rectangle and the type of the object is given on the top line of the rectangle.

The rectangle for the name for which the output is being generated is placed at the center of the diagram. All other objects are placed along the left and right margins if involved in "flow" relationships, and along the top or bottom margins if involved in "structure" or "updating" relationships. The type of relationship the center object has with bordering objects is given on the bottom line of the rectangle for each of the border objects. In the diagram, 'payroll-processing' RECEIVES 'employee-information,' etc.

In this example, names of objects have been shown in lower case letters and Types of Objects and Relationship in upper case letters. It is, therefore, easy to distinguish user assigned names for objects from words which are part of URL. The ability to distinguish lower and upper case letters depends on the facilities available in the installation in which URA is being used. If the installation does not support lower case letters, all words and names will appear in upper case.

University of Michigan - MTS

FORMATTED PROBLEM STATEMENT

```
PARAMETERS FOR: FPS

   FILE NOINDEX PRINT EMPTY NOPUNCH SMARG=5 MMARG=20 AMARG=10 EMARG=25 RMARG=70 CMARG=1 BMARG=40
   NODESIGNATE ONE-PER-LINE DEFINE COMMENT NONEW-PAGE NONEW-LINE NOALL-STATEMENTS
   COMPLEMENTARY-STATEMENTS LINE-NUMBERS PRINTEOF DLC-COMMENT

   1 INPUT                                          employee-information;
   2       /*   DATE OF LAST CHANGE - NOV 16, 1977, 08:32:38 */
   3      GENERATED BY:  departments-and-employees;
   4      RECEIVED BY:   payroll-processing;
   5
   6 INTERFACE                                      departments-and-employees;
   7       /*   DATE OF LAST CHANGE - NOV 16, 1977, 08:32:38 */
   8      GENERATES:     employee-information;
   9      RECEIVES:      paysystem-outputs;
  10
  11 OUTPUT                                         paysystem-outputs;
  12       /*   DATE OF LAST CHANGE - NOV 16, 1977, 08:32:38 */
  13      GENERATED BY:  payroll-processing;
  14      RECEIVED BY:   departments-and-employees;
  15
  16 PROCESS                                        payroll-processing;
  17       /*   DATE OF LAST CHANGE - NOV 16, 1977, 08:32:38 */
  18      GENERATES:     paysystem-outputs;
  19      RECEIVES:      employee-information;
  20      UPDATES:       payroll-master-information;
  21
  22 SET                                            payroll-master-information;
  23       /*   DATE OF LAST CHANGE - NOV 16, 1977, 08:32:38 */
  24      UPDATED BY:    payroll-processing;
  25
  26 EOF EOF EOF EOF EOF
```

Figure 1.2.6

University of Michigan - MTS

Process Picture

payroll-processing

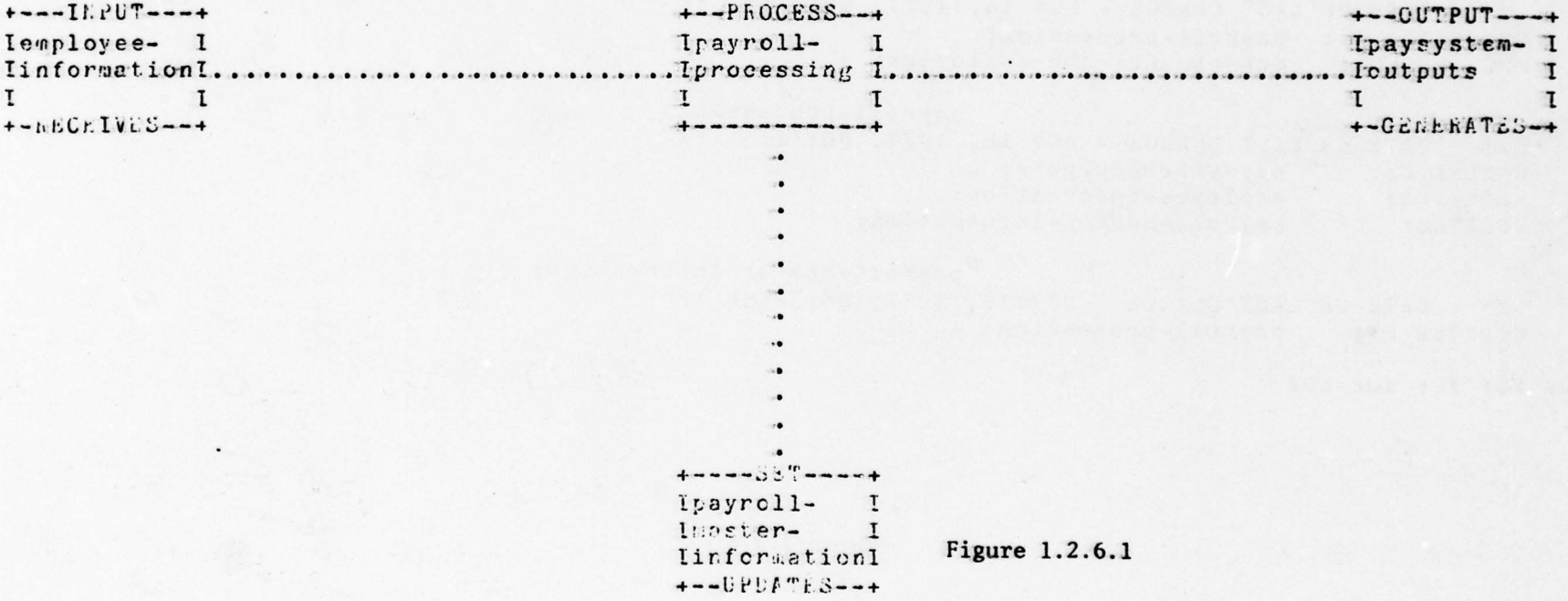


Figure 1.2.6.1

1.3 URL Objects

A URL object is anything given a URL name by the user of URL/URA. Each object is given a unique name so it can be identified each time it occurs in the system description. Consequently, all occurrences can be collected and analyzed. A URL name is one that conforms to the rules of name formation in the URL/URA system (Section 1.6). Once any particular object has been given a name it can be included in relationships only by specifying its name.

Each object must be a certain object type. The complete list of permissible types in alphabetical order is given in Figure 1.3 together with the allowable abbreviations for each object type. Of these, two are "special" types: SYNONYM and UNDEFINED. If the object type of an object is not declared explicitly, URA may be able to deduce the object type from the manner in which the object is used, otherwise, the object type for the name will be "UNDEFINED." A Problem Statement is not complete if it contains any UNDEFINED names. A SYNONYM is a special type of object that can be used only as an alias or pointer to one other name, e.g., an object that has been assigned the name 'validation-processing' might be given synonym 'valpr.'

Object Type	Abbreviation
ATTRIBUTE	ATTR
ATTRIBUTE-VALUE	ATTV
CLASSIFICATION	CLS
CONDITION	COND
ELEMENT	ELE
ENTITY	ENT
EVENT	EVT
GROUP	GP
INPUT	INP
INTERFACE	INTF, RWE, ORGU
INTERVAL	INT
KEYWORD	KEY
MAILBOX	BOX
MEMO	---
OUTPUT	OUT
PROBLEM-DEFINER	PD
PROCESS	PRC
PROCESSOR	PRCR
RELATION	RLN
RESOURCE	RSC
RESOURCE-USAGE-PARAMETER	RUP
SECURITY	SEC
SOURCE	SRC
SET	---
SUBSETTING-CRITERION	SSCN
SYNONYM	SYN
SYSTEM-PARAMETER	SYSP
TRACE-KEY	TKEY
UNDEFINED	---
UNIT	---

Figure 1.3 Object Types and Abbreviations

1.3.1 Classification of Object Types

For ease of describing the purpose and characteristics of each type of object with respect to the system documentation, it is convenient to group the types into classes. The list of classes and object types within each class if shown in Figure 1.3.1. It must be emphasized that classification is for exposition only and plays no role in the formal syntax or semantics of URL. The major categories of classification are the following:

Organization	for objects used to describe the organization or environment in which the target system is to operate.
Target System	for objects used to describe the target system.

Project Management	for objects used to describe the project developing the target system.
Properties	for objects used to describe the objects in the above three categories.

The purpose and characteristics of each object type is described below in the order in which listed in Figure 1.3.1. The relationships in which an object of a given type can be included is outlined in Section 1.4, and given in more detail in Sections 2 and 3. (The precise syntax is given in the "User Requirements Language, Language Reference Manual."[1]) A discussion of the role of each object type and situations in the system description process whether it should or should not be used is given in Section 4.

1.3.2 Organization Objects

The URL object used to describe some part of the organization or environment with which the target system interacts is called an INTERFACE (or REAL-WORLD-ENTITY). INTERFACES are often used to describe departments in an organization or other information processing systems which interface with the target system. Interfaces are sometimes called by such names as "stations," "organizational units," etc., in other documentation systems.

Interfaces are objects which, as far as the target system being developed, may receive data from it or transmit data to it. For example, if a warehouse stock control system were being designed, interfaces might be suppliers, customers, the accounting department, etc. They are not part of the target system, but have important relationships with it. Though the functions of an interface may be complex, only the description pertaining to its relationships with the target system are of importance. Interfaces should be described if they generate information to the target system, receive information from the target system, or are responsible for information within the target system.

---

[1] Part II of this document

CLASS OF OBJECT TYPES	OBJECT TYPES
INTERFACES OR ORGANIZATIONAL UNITS	INTERFACE (REAL-WORLD-ENTITY)
TARGET SYSTEM	
COLLECTIONS OF INFORMATION (EXTERNAL)	INPUT OUTPUT
(INTERNAL)	ENTITY
COLLECTIONS OF INFORMATION INSTANCES	SET
RELATIONSHIPS AMONG COLLECTION OF INFORMATION	RELATION
DATA DEFINITION	GROUP ELEMENT SUBSETTING-CRITERION
DATA DERIVATION	PROCESS
SIZE AND VOLUME	SYSTEM-PARAMETER INTERVAL
DYNAMIC BEHAVIOR	EVENT CONDITION
SYSTEM ARCHITECTURE	PROCESSOR RESOURCE RESOURCE-USAGE-PARAMETER UNIT
PROJECT MANAGEMENT	PROBLEM-DEFINER MAILBOX
PROPERTIES	SYNONYM KEYWORD ATTRIBUTE ATTRIBUTE-VALUE CLASSIFICATION MEMO SOURCE SECURITY TRACE-KEY
OTHER	UNDEFINED

Figure 1.3.1 Classification of Object Types

1.3.3 Target System Objects

Target system objects are used to describe the target system with respect to forms of information, processing of the information, behavior of the system over time, etc.

1.3.3.1 Collections of Information

Information related to, or pertaining to, one particular type of thing or concept is thought of as a collection of information. For example, "employee information" may be a collection of all information pertaining to a particular employee. This information would be derived when an employee is hired by the company, used to produce paychecks for the employee, updated to reflect changes in the employee's status, address, etc. The collection is to be thought of as a whole (in the above example, everything that had to be known about an employee) though in being processed by the target system, only portions of the collection might be used at any one time. There are three types of collections of information that may be defined in URL: INPUTS, OUTPUTS and ENTITIES. The difference among these types of collections is related to their role in the target system.

INPUTS

An INPUT is a collection of information produced external to the target system, but used by the target system. For example, in an inventory system, a customer order may be considered an INPUT to the system.

OUTPUTS

An OUTPUT is a collection of information produced by the target system, but which is used external to the system. For example, the paychecks produced by a payroll processing system could be thought of as OUTPUTS as they are eventually used by employees outside of the system. Once the collection has left the system, no further reference may be made to it.

ENTITIES

An ENTITY is a collection of information which is maintained internal to the system. ENTITIES are initially "produced" and then "maintained" using information from INPUTS and then OUTPUTS are produced using information from ENTITIES. The "employee information" described above in the definition of "Collections of Information" is an example of an ENTITY.

All of the above types of collections of information may belong to larger collections and may be broken down into smaller units

of information. Consequently, there may be "structural" relationships between particular objects of these types.

1.3.3.2 Collections of Information Instances

A number of instances of one or more collections of information is called a SET. For example, a SET might be defined to describe all instances of "employee information" in the target system. There is an important distinction to be made between a collection of information and an instance of this. Information called "employee information" is a collection of information, but employee information about JACK SMITH is an instance of the collection of information. A number of instances together may constitute a SET of "employee information." Likewise, if two collections of employee information were maintained (one for current employees and one for retired employees) a SET could be defined to contain instances of both collections as well as defining a separate SET for each collection of information about the different types of employees.

The common example of a SET is a master file consisting of records, i.e., ENTITIES, for each employee. However, SET may also consist of INPUTS and OUTPUTS. This permits SETS to represent collections of INPUTS, e.g., queues of messages to be processed.

1.3.3.3 Relationships Among Collections of Information

Collections of information maintained internal to the system (ENTITIES) are often "related" to each other in that there is information which is not inherent to either yet is associated with both. In the example of a warehouse stock control system, information about inventory items may be related to information about their suppliers, etc. RELATIONS are used to describe logical connections among ENTITIES.

1.3.3.4 Data Definition

Collections of information (INPUTS, OUTPUTS and ENTITIES) "contain" values of information called ELEMENTS and GROUPS.

ELEMENTS

ELEMENTS are the basic unit of information and, therefore, cannot be subdivided. An ELEMENT is used to describe a data object which may take on a value. For example, if "employee information" was defined to be an ENTITY it would not, in itself, have a value. The ELEMENTS making up "employee information" such as "age," "sex," "salary," etc., might have values for a particular instance of "employee information."

GROUPS

GROUPS are used to describe a collection of ELEMENTS and/or other GROUPS. GROUPS allow the problem definer to logically relate one or more ELEMENTS and/or GROUPS together and refer to them collectively by the GROUP name.

GROUPS can be thought to be synonymous with the names of the GROUP's components. In the example of "employee information," the "name" of the employee may be defined as a GROUP where the constitutents of the GROUP, "first name," "middle initial," "surname" may be defined as ELEMENTS. The use of GROUPS is primarily a notational convenience.

1.3.3.5 Data Derivation

An information processing system exists to process data, i.e., to produce data values from other data values. This transformation is known by different names such as process, procedure, function, operation activity, etc. In URL, a PROCESS is the type of object used to describe this transformation.

The total target system can be regarded as a PROCESS at the highest level. A PROCESS is defined by specifying the information upon which it operates and the information which it produces.

1.3.3.6 Size and Volume

Objects which relate information pertaining to the amount of information maintained by the system and volume of information to be processed are described to estimate the size of the target system.

Information about the size of a proposed target system is usually stated in terms of numbers. E.g., 500 employee changes occur each pay period or production analysis report consists of 100 pages.

In URL, the "parameters" affecting the size of the system are considered objects and each given a unique name: two types of objects are permitted:

SYSTEM-PARAMETERS and time INTERVALS

The basic purpose of treating these parameters as objects is that each occurrence can be uniquely identified. Consequently, all occurrences can be identified. Also, only one assignment of numerical values need be ready, the assignment can be as "late" as possible, and sensitivity-analysis can be carried out.

SYSTEM-PARAMETER

A SYSTEM-PARAMETER is used to represent a value relevant to characterizing "system" size. A SYSTEM-PARAMETER may be used to describe the number of instances of a particular ELEMENT in a particular instance of an ENTITY, for example.

INTERVAL

An INTERVAL is used to describe a unit of time. In defining frequency of an occurrence in the system, the frequency must be defined with respect to some unit of time. A "year" is an example of an interval, as is "work-week."

1.3.3.7 Dynamic Behavior

The description of the dynamic behavior of the system indicates requirements on processing order and the relationships between processes and objects that initiate, terminate, or interrupt them.

EVENT

An EVENT is used to describe possible occurrences during the operation of the target system. An occurrence of an EVENT is associated with a specific point in time, but the same EVENT may occur more than once during target system operation. For example, "error recognized" may be an EVENT that causes normal processing to be suspended while an error processor is initiated.

CONDITION

A CONDITION is used to describe some aspect of the state of the target system. A CONDITION may be either true or false. For example, "input data valid" could be a CONDITION. A change of this CONDITION from true to false might cause an EVENT (such as "error recognized") or might directly initiate error processing.

1.3.3.8 System Architecture Objects

PROCESSOR

An object that can "perform" a PROCESS is a PROCESSOR. In other words, a PROCESSOR is an "agent" that physically acts to perform a PROCESS. A computer system, a department in an organization, a person, can all be modeled as a PROCESSOR.

The total target system can be regarded as being performed by a single PROCESSOR at the highest level. This highest level PROCESSOR is the collection of all the physical entities (including human beings) that actually carries out all the information processing functions in the system.

RESOURCE

A RESOURCE is something that the physical elements in the target system consume in order to carry out information processing functions. A RESOURCE is consumed , and once an amount of RESOURCE is consumed, it is considered unrecoverable because it is "used up." For example, a certain amount of RESOURCE called electricity is consumed by an electrical appliance in a given time period. The amount of electricity thus consumed is not recoverable because it is used up.

It is important to note this somewhat specialized meaning of RESOURCE. In the general usage of the term, "resource" could mean something that is needed for a task to be performed, but which is returned after it is finished. For example, an electrical appliance can be regarded as a resource in this sense: when it is being used by someone, nobody else can use it; but when it is no longer used, it is available for use. In URL this second meaning of "resource" is modeled by PROCESSOR, and the term RESOURCE is exclusively used for the first meaning.

UNIT

Since it is necessary to handle quantities of RESOURCEs, units are needed to measure RESOURCEs. The object UNIT is used for this purpose. A UNIT is used to measure RESOURCEs. For example, electricity may be measured in a UNIT called "kilowatt-hour."

RESOURCE-USAGE-PARAMETER

A RESOURCE-USAGE-PARAMETER is an object that defines a measure of the RESOURCE usage for a PROCESS. It is introduced in URL as a way of expressing resource consumption of a PROCESSOR performing a PROCESS independent of what PROCESSOR performs it.

For example, one can assign values for a RESOURCE-USAGE-PARAMETER "no-of-fortran-steps" to a set of PROCESSes. The values of the RESOURCE-USAGE-PARAMETER for a PROCESS might signify the number of FORTRAN steps if the PROCESS is to be performed by a computer and FORTRAN is to be used to write the program. The actual amount of RESOURCE consumed in order to carry out this PROCESS depends on the particular PROCESSOR's ability, which is expressed in terms of RESOURCE consumption per RESOURCE-USAGE-PARAMETER.

1.3.4 Project Management Objects

Project management objects are used to provide information about the individual writing the URL description of the target system. URI is not intended to be a project management system, but it provides for two types of objects.

PROBLEM-DEFINER

PROBLEM-DEFINER is an object used to describe a person who writes the problem statement (URL statements) for the target system or who has the responsibility of maintaining the URL descriptions for one or more other URL objects. For example, PROBLEM-DEFINER, "Jane Smith," may be responsible for the URL description of the objects, "employee information," "payroll processing," etc., while other people on the project may be responsible for other objects in the target system's description.

MAILBOX

A MAILBOX is used to describe the location where questions and/or information about the URL description of a particular target system may be sent. Usually a MAILBOX is related to a PROBLEM-DEFINER.

1.3.5 Property Objects

For an accurate description of a target system, special properties of certain objects must be defined. For example, in describing a large information processing system, it may be necessary to define which functions (PROCESSES) are to be done manually, run batch, or on-line, etc. The URL object types that are available are SYNONYM, KEYWORD, ATTRIBUTE, ATTRIBUTE-VALUE, CLASSIFICATION, MEMO, SOURCE, SECURITY and TRACE-KEY.

SYNONYM

A SYNONYM is used to define an alternative name (alias) for a given name in the URL description of the system. The SYNONYM may simply define an abbreviation of a long name or specify a totally different name for an object, depending on who looks at the object (i.e., several people may think of the same thing, but call it several different names).

KEYWORD

A KEYWORD is an object type used to identify one or more objects in the target system description for selection and analysis

purposes. For example, if all functions (PROCESSES) described as being manual procedures in the target system were to be listed and analyzed together, the KEYWORD "manual" could be attached to each PROCESS for this purpose.

ATTRIBUTE and ATTRIBUTE-VALUE

ATTRIBUTES and ATTRIBUTE-VALUES are used to describe characteristics of particular objects in the target system description that may not be described by any other URL statements. For example, to describe that the length of an ELEMENT is six characters long, the ATTRIBUTE "length" could be defined and, for a particular ELEMENT, the corresponding ATTRIBUTE-VALUE could be "6."

CLASSIFICATION

CLASSIFICATION may be associated with data objects, PROCESSES and PROCESSORS in the target system. A value may also be associated with the CLASSIFICATION. In order for a PROCESS or PROCESSOR to be allowed access in the target system to a data object, it must have all the CLASSIFICATIONS that are associated with the data object, and the value must be greater than or equal to the value associated with the data object.

MEMO

A MEMO is used to describe a note (text) relevant to some aspect of the target system description. For example, a note concerning unresolved problems in describing a select number of GROUPS in the target system description could be defined as a MEMO and then related to each of the appropriate GROUPS.

SOURCE

A SOURCE is used to describe an object, outside of the problem statement, relevant to the description of one or more objects in the target system description. For example, a feasibility study of the target system being designed may have information relevant to why one alternative of describing the target system was chosen over another. The feasibility study could be designated as an object of type SOURCE.

SECURITY

SECURITY is used to identify what object descriptions can be reviewed by a given class of persons. Some types of information maintained by the target system may be considered confidential, so the description in the problem statement on how this

information is maintained may be restricted to high level management and a few select programmers.

TRACE-KEY

A TRACE-KEY is used to correlate objects which exist in different data bases. For example, the logical system design and physical system design of a security control system may exist in two different data bases. An object called a security level may exist in the logical design data base, and a field of numbers called a security level number may exist in the physical design data base. A TRACE-KEY called a security level key may be applied to both objects to display the correlation between them.

1.4 URL Relationships

The previous section presented the types of objects that must be defined when describing an information processing system. Organization objects define the environment in which the target system is embedded, Target System objects describe the components of the target system, Project Management objects describe the project in which the target system is being developed, and Property objects describe properties of all types of objects.

In addition to identifying particular objects (by giving them names), the relationships among these objects must be stated. For example, if "employee-information" is defined to be an INPUT and "payroll-processing" as a PROCESS, a relationship connecting these two objects may be specified. In URL terminology, if "employee-information" is an input to "payroll-processing," the relationship can be stated "payroll-processing" RECEIVES "employee-information" or "employee-information" is RECEIVED by "payroll-processing."

Figure 1.4 presents a listing of all relationships allowed in URL in alphabetical order along with legal abbreviations for these statements. (A dash in place of an abbreviation designates that there is no acceptable abbreviation for that statement.) This section gives an introduction to the relationships. They are defined in detail in Section 2 and 3.

1.4.1 Complementarity

One characteristic of most relationships between two names is that it may be specified in both directions. For example, specifying that an OUTPUT is GENERATED by a PROCESS is equivalent to specifying that the PROCESS GENERATES the OUTPUT. GENERATED and GENERATES are called complementary relationships. Figure 1.4.1 presents a list of all complementary relationship

pairs. (A dash designates that the relationship does not have a complement.)

1.4.2 Relationships Between Different Classes of Objects

URL allows a number of relationships to "connect" objects whether they are of the same class as defined in Section 1.3 or in different classes. For instance, in the above example, two Target System objects were related via the RECEIVES relationship. Since Organization objects, Project Management, and Property objects also contribute to the description of the system, they, too, must be related to defined Target System objects. Therefore, there is another set of relationships to connect Target System objects with Organization objects, another for Target System objects and Property objects, etc. The possible sets of relationships are shown in Figure 1.4.2.

Relationships may be classified in the same way as objects were classified in Section 1.2. The first row of Figure 1.4.3 presents relationships that an Organization object may have with other Organization objects, with Target System objects, Project Management objects and Property objects. The second row presents relationships that a Target System object may have with Organization objects, other Target System objects, Project Management objects and Property objects. The third row presents relationships that a Project Management object may have with Organization objects, Target System objects, other Project Management objects, and Property objects. The fourth row in Figure 1.4.2 presents relationships that a Property object may have with Organization objects, Target System objects, Project Management objects and other Property objects.

Relationship	Abbreviation	Relationship	Abbreviation
APPLIES	APP	PROCEDURE	PRCD
ASSERT	ASRT	RECEIVED	RCVD
ASSOCIATED	ASOC	RECEIVES	RCVS
ASSOCIATED-DATA	ASOD	RELATED	REL
ATTRIBUTES	ATTR	RESOURCE-USAGE	RU
BETWEEN	BTWN	RESOURCE-USAGE-PARAMETER-VALUE	RUPV
CARDINALITY	CARD	RESPONSIBLE	RESP
CAUSED	CSD	RESPONSIBLE-INTERFACE	RINT
CAUSES	CSS	RESPONSIBLE-PROBLEM-DEFINER	RPD
CLASSIFICATION	CLS	SECURITY	SEC
CONNECTIVITY	CONN	SECURITY-ACCESS-RIGHTS	SAR
CONSISTS	CSTS	SEE-MEMO	SM
CONSUMED	CNSD	SOURCE	SRC
CONSUMES	CNSS	SUBPARTS	SUBP
CONTAINED	CNTD	SUBSET	SST
DEPENDS	DPNS	SUBSETS	SSTS
DERIVATION	DRVN	SUBSETTING-CRITERIA	SSCA
DERIVED	DRVD	SUBSETTING-CRITERION	SSCN
DERIVES	DRVS	SYNONYM	SYN
DESCRIPTION	DESC	TERMINATED	TRMD
GENERATED	GEND	TERMINATES	TRMS
GENERATES	GENS	TERMINATION	TERM
HAPPENS	HAP	TERMINATION-CAUSES	TERC
IDENTIFIED	IDD	TRIGGERED	TRGD
IDENTIFIES	IDS	TRIGGERS	TRGS
INCEPTION	INCP	UPDATED	UPDD
INCEPTION-CAUSES	INCC	UPDATES	UPDS
INTERRUPTED	INTD	USED	--
INTERRUPTS	INTS	USES	--
KEYWORD	KEY	UTILIZED	UTLD
MADE	--	UTILIZES	UTLS
MAKES	MAK	VALUES	VAL
MAILBOX	BOX	VOLATILITY	VOL
MAINTAINED	MNTD	VOLATILITY-MEMBER	VOLM
MAINTAINS	MNTS	VOLATILITY-SET	VOLS
MEASURED	MSRD	WHILE	WHL
MEASURES	MSRS		
PART	--		
PERFORMED	PRMD		
PERFORMS	PRMS		

Figure 1.4
List of URL Statements in Alphabetical Order with Abbreviations

Relationship	Complementary Relationship
ASSERT	--
ASSOCIATED	ASSOCIATED-DATA
ATTRIBUTES	--
CARDINALITY	--
CAUSED	CAUSES
CONNECTIVITY	--
CONSUMED	CONSUMES
CONTAINED	CONSISTS
DEPENDS	--
DERIVED	DERIVES
GENERATED	GENERATES
HAPPENS	--
IDENTIFIED	IDENTIFIES
INCEPTION	INCEPTION-CAUSES
INTERRUPTED	INTERRUPTS
KEYWORD	APPLIES
MADE	MAKES
MAILBOX	APPLIES
MAINTAINED	MAINTAINS
MEASURED	MEASURES
PART	SUBPARTS
PERFORMED	PERFORMS
RECEIVED	RECEIVES
RELATED	BETWEEN
RESOURCE-USAGE	RESOURCE-USAGE-PARAMETER-VALUE
RESPONSIBLE-INTERFACE	RESPONSIBLE
RESPONSIBLE-PROBLEM-DEFINER	RESPONSIBLE
SECURITY	APPLIES
SEE-MEMO	APPLIES
SOURCE	APPLIES
SUBSET	SUBSETS
SUBSETTING-CRITERIA	SUBSETTING-CRITERION
SYNONYM	DESIGNATE
TERMINATED	TERMINATES
TERMINATION	TERMINATION-CAUSES
TRACE-KEY	APPLIES
TRIGGERED	TRIGGERS
UPDATED	UPDATES
USED	USES
UTILIZED	UTILIZES
VALUES	--

Figure 1.4.1
List of all URL Relationships with Complementary Relationships

	ORGANIZATION OBJECTS	TARGET SYSTEM OBJECTS
ORGANIZATION OBJECTS	SUBPARTS/PART	GENERATES RESPONSIBLE RECEIVES
TARGET SYSTEM OBJECTS	GENERATED RECEIVED RESPONSIBLE-INTERFACE	ASSOCIATED/ ASSOCIATED-DATA CARDINALITY CAUSED/CAUSES CLASSIFICATION CONNECTIVITY CONSUMED/CONSUMES CONTAINED/CONSISTS DERIVED/DERIVES GENERATED/GENERATES HAPPENS IDENTIFIED/IDENTIFIES INCEPTION/ INCEPTION-CAUSES INTERRUPTED/INTERRUPTS MADE/MAKES MAINTAINED/MAINTAINS MEASURED/MEASURES PART/SUBPART PERFORMED/PERFORMS RECEIVED/RECEIVES RELATED/RELATES RESOURCE-USAGE/ RESOURCE-USAGE- PARAMETER-VALUE SECURITY-ACCESS-RIGHTS SUBSET/SUBSETS SUBSETTING-CRITERIA/ SUBSETTING-CRITERION TERMINATED/TERMINATES TERMINATION/ TERMINATION-CAUSES TRACE-KEY TRIGGERED/TRIGGERS UPDATED/UPDATES UTILIZED/UTILIZES VALUES
PROJECT MANAGEMENT OBJECTS	RESPONSIBLE	RESPONSIBLE
PROPERTY OBJECTS	APPLIES	APPLIES

Figure 1.4.2 Relationships among Classes of Objects

	PROJECT MANAGEMENT OBJECTS	PROPERTY OBJECTS
ORGANIZATION OBJECTS	RESPONSIBLE-PROBLEM-DEFINER	ATTRIBUTES KEYWORDS SECURITY SEE-MEMO SOURCE SYNONYM TRACE-KEY
TARGET SYSTEM OBJECTS	RESPONSIBLE-PROBLEM-DEFINER	ATTRIBUTES KEYWORDS SECURITY SEE-MEMO SOURCE SYNONYM TRACE-KEY
PROJECT MANAGEMENT OBJECTS	MAILBOX/APPLIES	ATTRIBUTES KEYWORDS SECURITY SEE-MEMO SOURCE SYNONYM TRACE-KEY
PROPERTY OBJECTS	APPLIES	ATTRIBUTES KEYWORDS/APPLIES SECURITY/APPLIES SEE-MEMO/APPLIES SOURCE/APPLIES SYNONYM TRACE-KEY

Figure 1.4.2 Relationships among Classes of Objects (Continued)

1.4.3 Narrative and Text Description

Any information which is needed to describe an object and which cannot be specified by using one or more relationships can be specified in a narrative or text description called a comment entry. These comment entries are not named (as objects are name) and, therefore, apply to only one particular name. A number of different types of comment entries may be defined depending on the type of object they pertain to. The types of narrative and free-format descriptions that may be defined in URL according to the class of objects being described is given in Figure 1.4.3.

1.4.4 Classification of Relationships by System Aspects

The relationships may be grouped into nine major groups on the basis of the "aspect" of the system which they describe. These nine major aspects are:

System Flow
System Structure
Data Structure
Data Derivation
System Size and Volume
System Dynamics
System Architecture
System Properties
Project Management

Each is defined below. Specifying information about each of these aspects involves one or more object types and relationships. Figure 1.4.4 presents a summary of these nine aspects with corresponding objects and relationships.

1.4.4.1 System Flow

The System Flow aspect of the system deals with the interaction between the target system and its environment. This involves describing those objects (INPUTS) which are supplied by the environment (INTERFACES) to the target system, those objects (OUTPUTS) which are produced by the target system and accepted by the environment, and the responsibility of the environment for information (SETS) within the system.

Any transfers of data within the system are not considered as part of System Flow because there is no interaction with the environment.

1.4.4.2 System Structure

System Structure is concerned with the hierarchies inherent in most types of systems. (This includes information structures as well as processing structures.) Structures may also be introduced to facilitate a particular design approach such as "top down." In this context all information may initially be grouped together and called by one name at the highest level, and then gradually broken down. System structures can represent high-level hierarchies which may not actually exist in the system as well as those that do.

CLASS OF OBJECT TYPES	COMMENT ENTRY RELATIONSHIP
ORGANIZATION OBJECTS	DESCRIPTION
TARGET SYSTEM OBJECTS	DERIVATION DESCRIPTION PROCEDURE VOLATILITY VOLATILITY-MEMBER VOLATILITY-SET WHILE
PROJECT MANAGEMENT OBJECTS	DESCRIPTION
PROPERTY OBJECTS	DESCRIPTION

Figure 1.4.3 Types of Comment Entries for each Class of Objects

SYSTEM ASPECT	URL OBJECTS	URL RELATIONSHIPS
SYSTEM FLOW	INTERFACE	RECEIVES/RECEIVED[2]
	INPUT	GENERATES/GENERATED[2]
	OUTPUT	UPDATES/UPDATED[2]
	PROCESS	RESPONSIBLE-INTERFACE
	SET	
SYSTEM STRUCTURE	INTERFACE	SUBPARTS/PART OF
	INPUT	SUBSET/SUBSETS
	OUTPUT	UTILIZES/UTILIZED[2]
	PROCESS	
	SET	
	SUBSETTING-CRITERION	SUBSETTING-CRITERIA/SUBSETTING-CRITERION
DATA STRUCTURE	GROUP	CONSISTS/CONTAINED
	ELEMENT	IDENTIFIES/IDENTIFIED
	ENTITY	
	RELATION	
	CLASSIFICATION	CLASSIFICATION ASSOCIATED/ASSOCIATED-DATA
DATA DERIVATION	INTERFACE	USES/USED
	INPUT	DERIVES/DERIVED[2]
	OUTPUT	UPDATES/UPDATED[2]
	PROCESS	MAINTAINS/MAINTAINED[2]
	SET	PROCEDURE [1]
	GROUP	DERIVATION [1]
	ELEMENT	
	ENTITY	SECURITY-ACCESS-RIGHTS
SYSTEM SIZE	SYSTEM-PARAMETER	CONSISTS
	INTERVAL	HAPPENS
		CONNECTIVITY
		CARDINALITY
		VALUES
		VOLATILITY [1]
		VOLATILITY-SET [1]
		VOLATILITY-MEMBER [1]

[1] comment entry
[2] conditional (DEPENDING ON) and repetition (FOR EACH) clauses are allowed

Figure 1.4.4
URL object and Statements Organized According to Aspect of Target System Described

SYSTEM ASPECT	URL OBJECTS	URL RELATIONSHIPS
SYSTEM DYNAMICS	EVENTS	CAUSES/CAUSED[2]
	CONDITION	INCEPTION-CAUSES/ ON INCEPTION[2]
		INTERRUPTS/INTERRUPTED[2]
		MAKES/MADE
		TERMINATES/TERMINATED[2]
		TERMINATION-CAUSES/ ON TERMINATION[2]
		TRIGGERS/TRIGGERED[1]
		WHILE [1]
SYSTEM ARCHITECTURE	PROCESSOR	CONSUMES/CONSUMED
	RESOURCE	PERFORMS/PERFORMED
	RESOURCE-USAGE-PARAMETER	RESOURCE-USAGE/ RESOURCE-USAGE-PARAMETER-VALUE
	UNIT	MEASURES/MEASURED
PROJECT MANAGEMENT	PROBLEM-DEFINER	RESPONSIBLE/ RESPONSIBLE-PROBLEM-DEFINER
	MAILBOX	MAILBOX/APPLIES
PROPERTIES	ATTRIBUTE/ ATTRIBUTE-VALUE	ATTRIBUTES/APPLIES
		KEYWORDS/APPLIES
		SEE-MEMO/APPLIES
	KEYWORD	SYNONYMS/DESIGNATE
	MEMO	DESCRIPTION [1]
	SYNONYM	SOURCE/APPLIES
	SOURCE	SECURITY/APPLIES
	SECURITY	TRACE-KEY/APPLIES
	TRACE-KEY	ASSERT

[1] comment entry
[2] conditional (DEPENDING ON) and repetition (FOR EACH) clauses are allowed

Figure 1.4.4 (Continued)
URL Object and Statements Organized According to Aspect of Target System Described

1.4.4.3 Data Structure

Data Structures represent the relationships that exist among data used and/or manipulated by the system as seen by the "users" of the system. Data Structures also exist in the way data is grouped in collections of information such as documents. The description of Data Structures also involves specification of relationships among logical collections of data and the data associated with such relationships.

1.4.4.4 Data Derivation

The Data Derivation aspect of the system description specifies the way in which data is manipulated or derived by the system. It specifies what information is used, updated and/or derived, how this is done, and by which processes. This aspect differs from System Flow, since System Flow only designates the inputs to th system and the end results (OUTPUTS), without specifying what actions take place to bring these transformations about. Data Derivation can deal with the very lowest transformations of data, whereas, System Flow deals with high level collections of information (i.e., INPUTS and OUTPUTS).

1.4.4.5 System Size

The System Size is concerned with the size of the system and those factors that influence the volume of processing that will be required. To describe system size, the parameters involved are named as objects.

1.4.4.6 System Dynamics

The dynamic analysis aspect of system description presents the manner in which the target system behaves over time. EVENTS and CONDITIONS are used to help describe when PROCESSES are performed and under what conditions.

1.4.4.7 System Architecture

The System Architecture aspect deals with the physical components and structures that are necessary in order to realize the given user requirements.

1.4.4.8 Project Management

In addition to the description of the target system being designed, documentation of the group designing (or documenting) the target system is needed. This involves identification of persons involved and their responsibilities, etc.

1.4.4.9 Properties

All objects (of a particular type) used to describe the target system have characteristics that distinguish them from other objects of the same type. Therefore, the properties of particular objects in the system must be described. In general, Properties involve any description particular to a given object.

1.5 System Documentation Using URL/URA

The Process of using URL/URA to describe an information processing system includes the following steps:

1) Gathering information about the system.
2) Expressing the information in URL.
3) Formatting URL as required by URA.
4) Converting the Problem Statement into computer processable form.
5) Entering the data into the project data base.
6) Generating outputs from the data base.

1.5.1 Gathering Information About the System

This step can be carried out as with present manual methods. However, the URL structure (sections and statements) can be used as a structure for which information is collected. URA outputs can also be used as a checklist of missing information.

1.5.2 Expressing the Information in URL

The use of URL consists of:

- Identifying objects, naming them and assigning a unique type to each.
- Determining the relationships among the objects.
- Stating appropriate properties for each object.

Any information which cannot be expressed in this formal syntax can be given as text in comment entries.

1.5.3 Formatting URL as Required by URA

URA requires only minimal formatting as described in Section 1.6.

1.5.4 Converting the Problem Statement into Computer Processable Form

The problem statement can be read by URA from whatever form of computer processable input is desired. The usual procedure will be to punch the problem on cards or to enter it via a terminal.

The data can be entered on cards anywhere in the first 72 columns.

When data is entered via a terminal, it will normally first be entered into a file.

1.5.5 Entry of the Data into the Project Data Base

In ISDOS terminology, the description of a proposed Information Processing System is called a Problem Statement in the sense that it represents a "problem" to be solved. The physical system designer then has the problem of finding the best system to accomplish the requirements implicit in the description of the proposed Information Processing System. (The proposed system can be considered a solution to an earlier problem, namely, the problem of what outputs are necessary to satisfy the "users" needs for information.) The URA data base contains the problem statement as it has been given up to that time.

The problem statement will be built up over a period of time, possibly by a number of problem definers working simultaneously. Three aspects of a problem statement and its use during logical system design need to be considered:

1) The documentation of the problem statement available to the problem definer based on the URL information in the data base.

2) The problem statement as it exists in the data base.

The data base contains information about all the objects that have been identified, and all the relationships among those objects that have been specified. It also contains narrative statements to be used in the final documentation. Except for the narrative statements, the data is stored in "coded" form and not as a copy of the FORMATTED PROBLEM STATEMENT.

3) The method by which the problem statement is added to or modified.

When the problem definer wishes to add to the data base or modify it in some way, input is prepared according to the syntax of URL, i.e., in the same form as the FORMATTED PROBLEM STATEMENT. However, only new data or changes to the data need to be entered. Any data not affected by the input will remain unchanged.

The use of URL, therefore, differs from present methods in two significant ways: the information about a proposed system can be entered in any order and only new data or changes need be entered.

1.5.6 Generating Outputs from the Data Base

At any time problem definers can obtain outputs based on all or part of the data in the data base. These outputs would be used by the problem definers in their own work (Data Collection, Analysis, Design, Evaluation or Improvement) or in conferring with users and others. The complete statement containing all the data in the data base is called the FORMATTED PROBLEM STATEMENT. The other outputs contain subsets of the total documentation, summaries, rearrangements and analyses. The URA User's Manual gives a complete description of each report available to the problem definer.

The FORMATTED PROBLEM STATEMENT is based on all the data in the data base. It is not merely a listing of the data that has been entered but includes, in addition to the relationships explicitly stated, those that have been implied (i.e., complementary relationships). The FORMATTED PROBLEM STATEMENT is "syntactically" correct, i.e., it can be processed by URA. In practice problem definers would use the FORMATTED PROBLEM STATEMENT in conjunction with other reports from URA.

1.6 User Requirements Language: Syntax Structure

Since URL is a language which must be understood by a computer program, (URA), it must have a formal structure, usually referred to as syntax. In this section, the syntax structure of URL is outlined. A more detailed statement of the syntax for URL appears in "User Requirements Language, Language Reference Manual."[1]

1.6.1 Language Structure

URL consists of several levels:

Syntax Level	URL Description Constitutents
1	Target System Description
2	URL Section
3	URL Statements
4	Reserved Words, Names, Numbers
5	Characters

---

[1] Part II of this document.

A description at each level consists of one or more units of the succeeding levels. A system description consists of one or more URL sections. A URL section consists of one or more URL statements. Each URL statement is formed by some combination of Reserved Words, Names, and/or Numbers. Finally, the Reserved Words, Names and Numbers consist of characters allowed by the URL character set.

The syntax of the constituents at each level are defined in the remainder of this section.

1.6.2 Problem Statement Format

URL is a free-format language in contrast to "fixed-format" or "tabular." In particular, this means that URL descriptions can appear anywhere on the physical medium, such as punched cards and that within fairly wide limits, information can be entered in any order.

The program which "reads" the Problem Description understands or decodes the descriptions by reorganizing a delimiter the semi-colon (;) and Reserved Words. The latter are defined in 1.6.6.

The major advantages of free format are that complex problem statements can be made with relative ease and the problem statements can be made fairly concise.

Forms can be designed if a more structured method of recording the problem statement is required. One possible organization of the forms is given in Section 3.

1.6.3 Target System Identification

Only one URA data base is needed to store all information about a given System. This data base represents the up-to-date version of the system description. URA has facilities for specifying the name of the system being described on all reports generated from the data base.

1.6.4 URL Sections

A URL description or Problem Statement consists of one or more URL sections. Each section consists of one or more URL statements. The first statement in a section (and the only required one) is called the Section Header. A Section Header is a URL statement that identifies a section and specifies:

1) That the user defined names given in the section header are a particular object type (e.g., PROCESS or SET, etc.).

2) That any URL statements (up to the next Section Header) present some description about the name(s) given in the header and/or form relationships between names in the header and other user-defined names in the problem statement.

There are a finite number of URL statements that are defined as Section Headers and are given in Table 1.6.4.

CONDITION	MEMO
DEFINE	OUTPUT
DESIGNATE	PROBLEM-DEFINER
ELEMENT	PROCESS
ENTITY	PROCESSOR
EVENT	RELATION
GROUP	RESOURCE
INPUT	RESOURCE-USAGE-PARAMETER
INTERFACE	SET
INTERVAL	UNIT

Table 1.6.4. Section Header Statements

Most object types are defined in sections of the same time, i.e., a PROCESS would be defined in the PROCESS section. Therefore, there is a one to one correspondence between types of objects and section headers except that the following types of objects are all defined in a DEFINE section:

ATTRIBUTE	SECURITY
ATTRIBUTE-VALUE	SOURCE
CLASSIFICATION	SUBSETTING-CRITERION
KEYWORD	SYSTEM-PARAMETER
MAILBOX	TRACE-KEY

and a SYNONYM is assigned to an object by a DESIGNATE section. This distinction between Type of Object and Section Header is immaterial conceptually and is introduced only to simplify entering URL information into the data base since all the Types of Objects described in a DEFINE section allow essentially the same set of statements.

For each type of section header there are a finite number of different URL statements that can be specified after it. For example, if the section defines a name to be an INPUT it may be desirable to say what GENERATES and what RECEIVES the INPUT, but it would be illogical to say that the INPUT MAINTAINS other information. Therefore, there are a select set of URL statements that may be used in conjunction with a particular Section Header. The Section Summaries in Section 3 and in "User Requirements Language, Language Reference Manual,"[1] present a list of which URL statements which can appear in each type of

---

[1] Part II of this document.

section.

1.6.5 URL Statements

There are three basic types of URL statements:

1) Section Header statement - This type of statement is used to define one or more names (objects) to be a particular object type (e.g., PROCESS or GROUP) as described above.

2) Relationship statement - This type of statement is used to specify relationships between or among objects. In specifying the target system description, it is necessary to describe which INTERFACES supply which INPUTS to which PROCESSES, what data (GROUPS and ELEMENTS) are used by what PROCESSES, what EVENTS cause which PROCESSES to be triggered, and how often, etc. For each type of object particular relationships can be specified as outlined in 1.4 and described in more detail in Section 2 and 3. For example, a relationship between an OUTPUT and the PROCESS which produces the OUTPUT would be specified by the GENERATES statement. A PROCESS can GENERATE an OUTPUT. Likewise, an OUTPUT may be GENERATED by a PROCESS.

3) Comment Entry statement - This type of statement is used to relate a narrative (or text) description (comment entry) to a particular object. Text descriptions, therefore, may be used to supplement relationships; this means that any information which cannot be directly specified in one or more relationships (relationship statements) can be presented in a narrative format.

All URL statements begin with a reserved word and are terminated by a semi-colon (;). If the statement specifies a relationship (one of the types of statements defined previously) then the statement must also consist of one or more user defined names and may not require one or more reserved words. Optional words may be inserted in the statements. For example:

RECEIVED BY employee-processing;

begins with the reserved words, RECEIVED, which is followed by an optional word, BY, then by a user-defined name, "employee-processing" and finally, terminated by a semi-colon. Blanks are used to separate words and names in the statement. Any number of blanks may be used where one blank is allowed.

If the statement is a comment entry type statement, then the first line of the statement may only consist of reserved words and the semi-colon. Succeeding lines of the statement are interpreted as the comment entry text until a semi-colon is encountered. Therefore, a semi-colon may not be used in the

text of a comment entry statement. For example:

```
DESCRIPTION;
     The time card is the record of hours an hourly
     employee worked in any given week.  ;
```

Any characters, except the semi-colon, may appear in the text of a comment entry such as the period (.) in this example. The comment entry text may not begin on the same line as the reserved word for the statements.

In many statements which specify relationships among objects, a list of user defined names may be given. For example:

```
USES:  fica-tax, federal-tax, state-tax;
```

designates that the names in the section header, to which this statement belongs, USE fica-tax, federal-tax and state-tax. Blanks may be used on either side of the commas separating the user defined names.

Abbreviations of reserved words may be used in place of the full reserved word. For example:

```
RECEIVED BY:  employee-processing;
```

may also be given as:

```
RCVD employee-processing;
```

The allowable abbreviations (which are also designated as being reserved words) are given in Appendix D of "The User's Requirements Language, Language Reference Manual."[1]

1.6.6 Reserved Words, Names and Numbers

Reserved words are combinations of letters and dashes used to identify URL section headers, URL statements and optional words. There is a limited number of reserved words as given by Appendix B of "The User's Requirements Language, Language Reference Manual."[1] All reserved words are defined by the URL/URA system and may not be changed by the user.

Optional words may be used by the problem definer to improve the readability of the problem statement. Words like BY, A, ARE, AND, etc. are legal URL optional words. Appendix C of "The User Requirements Language, Language Reference Manual" is a list of all URL optional words. In the following URL statement:

```
USED BY employee-processing TO DERIVE paycheck;
```

---

[1] Part II of this document.

the words, USED, BY, TO and DERIVE are URL reserved words.

User defined names are any names (words) used in a URL statement that are not URL reserved words. Restrictions on user defined names are that they may only begin with a letter, consist of only letters, digits and dashes, and be no longer than thirty characters in length. The names "employee-processing" and "paycheck" in the previous example are instances of user defined names.

Numbers used in a URL statement may only consist of the digits 0 through 9 with no decimal points plus or minus signs, etc., allowed.

1.6.7 Character Set

All Reserved words, names and numbers must be composed of characters in the URL character set. The attached list gives for each ASCII character a code of 1 to 4 classifying the characters into the following categories:

Code 1: Nonprinting operating System and transmission control characters to be treated as punctuation, but will always be illegal.

Code 2: Puncutation, delimiters, etc. which are not allowed in names.

Code 3: Characters allowed at any position in a name.

Code 4: Characters allowed at any position in a name after the first.

There are three versions of this categorization:

1. A one page summary.
2. Sorted by Octal representation.
3. Sorted by code, then by Octal representation.

CODE 1: All others

CODE 2: "&'()*,:;=?[ ]|~

CODE 3: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
!#$%@{}\^`

CODE 4: 0123456789
+-./<>_

CODE	OCTAL	CHAR	NAME
1	000	nul	null or time fill char
1	001	soh	start of heading
1	002	stx	start of text
1	003	etx	end of text (EOM)
1	004	eot	end of transmission (EOT)
1	005	enq	enquiry (WRU)
1	006	ack	acknowledge (RU)
1	007	bel	bell
1	010	bs	backspace
1	011	ht	horizontal tabulation (TAB)
1	012	lf	line feed (LINE FEED)
1	013	vt	vertical tabulation (VT)
1	014	ff	form feed (FORM)
1	015	cr	carriage return (RETURN)
1	016	so	shift out
1	017	si	shift in
1	020	dle	data link escape
1	021	dc1	device control 1 (X-ON)
1	022	dc2	device control 2 (TAPE)
1	023	dc3	device control 3 (X-OFF)
1	024	dc4	device control 4 (TAPE)
1	025	nak	negative acknowledge
1	026	syn	synchronous idle
1	027	etb	end of transmission blocks
1	030	can	cancel
1	031	em	end of medium
1	032	ss	special sequence
1	033	esc	escape
1	034	fs	file separator
1	035	gs	group separator
1	036	rs	record separator
1	037	us	unit separator

CODE	OCTAL	CHAR	NAME
2	040	sp	space (SPACE BAR)
3	041	!	exclamation point
2	042	"	quotation mark
3	043	#	number sign
3	044	$	currency symbol
3	045	%	percent
2	046	&	ampersand
2	047	'	apostrophe
2	050	(	opening parenthesis
2	051	)	closing parenthesis
2	052	*	asterisk
4	053	+	plus
2	054	,	comma
4	055	-	hyphen or minus
4	056	.	period
4	057	/	slant
4	060	0	zero
4	061	1	one
4	062	2	two
4	063	3	three
4	064	4	four
4	065	5	five
4	066	6	six
4	067	7	seven
4	070	8	eight
4	071	9	nine
2	072	:	colon
2	073	;	semicolon
4	074	<	less than
2	075	=	equal
4	076	>	greater than
2	077	?	question mark

CODE	OCTAL	CHAR	NAME
3	100	@	commercial at
3	101	A	
3	102	B	
3	103	C	
3	104	D	
3	105	E	
3	106	F	
3	107	G	
3	110	H	
3	111	I	
3	112	J	
3	113	K	
3	114	L	
3	115	M	
3	116	N	
3	117	O	
3	120	P	
3	121	Q	
3	122	R	
3	123	S	
3	124	T	
3	125	U	
3	126	V	
3	127	W	
3	130	X	
3	131	Y	
3	132	Z	
2	133	[	opening bracket
3	134	\	reverse slant
2	135	]	closing bracket
3	136	^	circumflex
4	137	_	underline

CODE	OCTAL	CHAR	NAME
3	140	`	grave accent
3	141	a	
3	142	b	
3	143	c	
3	144	d	
3	145	e	
3	146	f	
3	147	g	
3	150	h	
3	151	i	
3	152	j	
3	153	k	
3	154	l	
3	155	m	
3	156	n	
3	157	o	
3	160	p	
3	161	q	
3	162	r	
3	163	s	
3	164	t	
3	165	u	
3	166	v	
3	167	w	
3	170	x	
3	171	y	
3	172	z	
3	173	{	opening brace
2	174	|	vertical line
3	175	}	closing brace
2	176	~	tilde
1	177	del	delete (RUBOUT)

CODE	OCTAL	CHAR	NAME
1	000	nul	null or time fill char
1	001	soh	start of heading
1	002	stx	start of text
1	003	etx	end of text (EOM)
1	004	eot	end of transmission (EOT)
1	005	enq	enquiry (WRU)
1	006	ack	acknowledge (RU)
1	007	bel	bell
1	010	bs	backspace
1	011	ht	horizontal tabulation (TAB)
1	012	lf	line feed (LINE FEED)
1	013	vt	vertical tabulation (VT)
1	014	ff	form feed (FORM)
1	015	cr	carriage return (RETURN)
1	016	so	shift out
1	017	si	shift in
1	020	dle	data link escape
1	021	dc1	device control 1 (X-ON)
1	022	dc2	device control 2 (TAPE)
1	023	dc3	device control 3 (X-OFF)
1	024	dc4	device control 4 (TAPE)
1	025	nak	negative acknowledge
1	026	syn	synchronous idle
1	027	etb	end of transmission blocks
1	030	can	cancel
1	031	em	end of medium
1	032	ss	special sequence
1	033	esc	escape
1	034	fs	file separator
1	035	gs	group separator
1	036	rs	record separator
1	037	us	unit separator
1	177	del	delete (RUBOUT)

CODE	OCTAL	CHAR	NAME
2	040	sp	space (SPACE BAR)
2	042	"	quotation mark
2	046	&	ampersand
2	047	'	apostrophe
3	050	(	opening parenthesis
3	051	)	closing parenthesis
2	052	*	asterisk
2	054	,	comma
2	072	:	colon
2	073	;	semicolon
2	075	=	equal
2	077	?	question mark
2	133	[	opening bracket
2	135	]	closing bracket
2	174	|	vertical line
2	176	~	tilde

CODE	OCTAL	CHAR	NAME
3	041	!	exclamation point
3	043	#	number sign
3	044	$	currency symbol
3	045	%	percent
3	100	@	commercial at
3	101	A	
3	102	B	
3	103	C	
3	104	D	
3	105	E	
3	106	F	
3	107	G	
3	110	H	
3	111	I	
3	112	J	
3	113	K	
3	114	L	
3	115	M	
3	116	N	
3	117	O	
3	120	P	
3	121	Q	
3	122	R	
3	123	S	
3	124	T	
3	125	U	
3	126	V	
3	127	W	
3	130	X	
3	131	Y	
3	132	Z	

CODE	OCTAL	CHAR	NAME
3	134	\	reverse slant
3	136	^	circumflex
3	140	`	grave accent
3	141	a	
3	142	b	
3	143	c	
3	144	d	
3	145	e	
3	146	f	
3	147	g	
3	150	h	
3	151	i	
3	152	j	
3	153	k	
3	154	l	
3	155	m	
3	156	n	
3	157	o	
3	160	p	
3	161	q	
3	162	r	
3	163	s	
3	164	t	
3	165	u	
3	166	v	
3	167	w	
3	170	x	
3	171	y	
3	172	z	
2	173	{	opening brace
2	175	}	closing brace

CODE	OCTAL	CHAR	NAME
4	053	+	plus
4	055	-	hyphen or minus
4	056	.	period
4	057	/	slant
4	060	0	zero
4	061	1	one
4	062	2	two
4	063	3	three
4	064	4	four
4	065	5	five
4	066	6	six
4	067	7	seven
4	070	8	eight
4	071	9	nine
4	074	<	less than
4	076	>	greater than
4	137	_	underline

The one exception to this is that any characters may be used in the text of a comment entry statement.

1.6.8 Format Restrictions

While URL is a free format language, there are certain restrictions that have been incorporated into the implementation of URA to facilitate entry of Problem Statements.

One restriction is concerned with length of the statement. Though a statement may extend over any number of lines, only the first 72 columns of a card, or characters in a message of each line may be used. Anything over this will be ignored. Therefore, the statement:

```
RECEIVED BY:  employee-processing;
```

may also be given as:

```
RECEIVED
BY
:
 employee-processing
;
```

with no effect on how this statement is interpreted by URA. The only restriction is that the statement may only be split where a blank is allowed and not in the middle of a reserved word or user defined name.

A second restriction is the one mentioned above for comment entries. The type of Comment Entry such as DESCRIPTION or PROCEDURE must appear on a separate line, followed by the text ending in a semi-colon.

A third restriction is the use of EOF as a special type of statement that designates the end of a collection of URL sections to be used as input to URA. This statement specifies that there are no more URL statements following and that URA may stop processing of the URL statements. The EOF statements must be used whenever URL statements are given as input to the INPUT-PSL or DELETE-PSL commands in URA.

1.7 Comparison of Manual and Computer-Aided Documentation in Logical Systems Design

1.7.1 Description in a Structured Language Compared to Manual Methods Using Narrative, Forms and Charts

A number of desirable properties of documentation were outlined in Section 1.1. The present manual and computer-aided methods may be compared, as follows:

Present Manual Documentation	Computer-Aided Documentation
Hard to Understand	Understandable
Ambiguous	Precise
Inconsistent	Consistent
Incomplete	Complete
Incorrect	Correct
Difficult to Analyze and Evaluate	Computer-Aided Analysis and Evaluation
Hard to Modify	Computer-Aided-Updating

A more comprehensive description of how desirable characteristics of computer-aided documentation can be achieved is given in Section 5. The contribution of the structured description language is outlined in 1.7.2 and the contribution of the outputs available for URA is outlined in 1.7.3.

1.7.2 The Advantage of a Structured Description Language

The major characteristics of URL for describing systems are:

1) Each object has a unique name.
2) Each relationship has a precise format, i.e., syntax.
3) Only a specified number of relationships may exist among objects of given types.
4) Any number of properties may be defined for objects of a given type but each property must be uniquely named.

The differences between URL and the usual method of

documentation with narrative text and manual flow charts are shown in the following table:

	Narrative	URL
Object Names	unlimited	unique
Number of objects	unlimited	essentially unlimited - limited only because names must not be more than 30 characters and the first letter character must be a letter
Type of objects	not necessarily stated	relatively small number of explicitly defined types
Relationships	unlimited and not necessarily explicitly defined	relatively small number of explicity defined types
Properties	unlimited and not necessarily explicitly defined	unlimited but explicitly defined

1.7.3 Outputs Available from URA

URA provides a number of standard outputs which can be used to satisfy the documentation requirements for aiding the:

Problem Definer in His Own Work
Problem Definer in Communication with Users
Coordination in Project
Final Documentation

1.7.4 Changes in Logical Design

The use of a computer-aided system allows changes in the way logical design is carried out. Table 1.7.4 summarizes the differences between the manual and computer-aided methods and the resulting improvements in the various logical system design activites: data collection, analysis, design, evaluation and improvement.

	Difference Between Manual and Computer-Aided Methods
Data Collection	Forms of standard URL format can be used to record collected data.
Analysis	Analyses for correctness, completeness and consistency of data are done when inputting data to URA and on demand from URA.
Design of Proposed System	Though design is a creative process, URA can make more data available to the designer and in a formatted matter.
Evaluation	URA generates accurate, standard reports to aid in the evaluation process.
Improvement	Modification of the problem statement is easily made through availability of data base modification commands.

	Improvement in Computer-Aided Methods
Data Collection	Outputs from URA can provide a checklist for deciding what additional information is needed.
Analysis	Use of the "URA data base" insures that analysis is always performed on an up-to-date version of the problem statement. As new analysis methods are developed, they can be incorporated into URA.
Design of Proposed System	Use of the URA reports allows the designer to look at particular aspects of the system of interest. Simple modifications to the data base can present alternatives in design.
Evaluation	URA provides some rudimentary facilities for computing volume or work measures from the data in the problem statement. As additional methods are developed, they can be incorporated.
Improvement	Rather than "starting from scratch" to incorporate changes in the problem statement, improvements can be made on the original URA data base.

Table 1.7.4
Changes in Logical Design Procedure and Value of Change

2. PROBLEM STATEMENT FACILITIES BY SYSTEM ASPECT

To accurately describe a system it is necessary to describe all aspects identified in 1.4. The following sections present the URL objects and URL statements that pertain to each aspect of the system description:

System Flow
System Structure
Data Structure
Data Derivation
System Size
System Dynamics
System Architecture
System Properties
Project Management

Guidelines are also provided to aid the analyst in describing a particular system in URL, including guidelines to help map the objects, as they exist in the real world, into what they may be called in URL terminology. The Analyzer outputs relevant to each aspect of the description are also presented to aid the analyst in making the description consistent and complete.

The explanations of URL statements are given at three levels of precision:

"must" - denotes that this is checked by URA and not entered into the data base unless correct. Note the "must" does not necessarily imply in this sense that the particular statement has to be in the data base.

"can" - denotes that a choice is available. Each choice selected is checked by URA and not entered into the data base unless correct.

"should" - denotes that this is not checked by URA before stored in the data base but is necessary for a complete description of the target system. Some of these "completeness" checks are made when producing URA reports and warning messages are produced. Others can be made by the analyst using URA reports.

"implies" and "may" - denotes the semantic meaning of the statement. This is not checked by URA nor necessary for a complete description. Interpretation is to be decided by the Problem Definer and organization.

2.1 System Boundaries and Input Output Flow

One URA data base describes one Information Processing System and objects associated with it. The description of a system can begin by describing its boundaries. (Identifying the boundary of a system is not always easy; considerations involved in this process are discussed in 4.1.) This section describes the URL facilities in specifying system boundaries and flow to and from the system.

2.1.1 System Flow Objects

The boundary of the target system is described in terms of the objects which flow across the boundaries.

INPUT - an object which contains data and flows into the target system from an external object (i.e., INTERFACE) to an internal object, (i.e., a PROCESS).

OUTPUT - an object which contains data and flows from the target system to an external object from an internal object (i.e., a PROCESS) to an external object (i.e., an INTERFACE).

SET - an object which designates a collection of data containers and is stored and updated by an internal object, (i.e., PROCESS).

INTERFACE (or REAL-WORLD-ENTITY) - an external object which can produce an INPUT, receive an OUTPUT or be RESPONSIBLE FOR a SET.

PROCESS - an internal object which can accept an INPUT or produce an OUTPUT or UPDATE a SET.

2.1.2 System Flow Relationships

The verbs in the above definitions that are formal URL relationships are:

GENERATES/GENERATED BY

An INTERFACE must GENERATE an INPUT or the INPUT must be GENERATED BY an INTERFACE. A PROCESS must GENERATE an OUTPUT or the OUTPUT must be GENERATED BY a PROCESS.

RECEIVES/RECEIVED BY

An INTERFACE must RECEIVE an OUTPUT or the OUTPUT must be RECEIVED BY an INTERFACE. A PROCESS can RECEIVE an INPUT or the

INPUT can be RECEIVED BY a PROCESS.

UPDATES/UPDATED BY

A PROCESS must UPDATE a SET or the SET must be UPDATED BY a PROCESS.

Conditional actions can be described by the use of the DEPENDING ON clause. Also, repetition of actions can be described by the use of the FOR EACH clause.

RESPONSIBLE FOR/RESPONSIBLE-INTERFACE

An INTERFACE must be RESPONSIBLE FOR a SET. A SET must have a RESPONSIBLE-INTERFACE.

2.1.3 System Flow Syntax and Semantics

The objects and relationships involved in describing system flow are shown pictorially in Figure 2.1.3 and in tabular form in Table 2.1.3. The direction for reading the table is from the left object to the desired relationship and then up to the particular object.

An INTERFACE can GENERATE any number of INPUTS, RECEIVE any number of OUTPUTS, and be RESPONSIBLE for any number of SETS.

An INPUT can be GENERATED by any number of INTERFACES (implies any one of them must GENERATE it) and be RECEIVED BY more than one PROCESS (implies that all of them must RECEIVE it).

A SET may have any number of RESPONSIBLE-INTERFACES (this implies that all are) and may be UPDATED by any number of PROCESSES (implies that all do).

2.1.4 System Flow Common Equivalents and Usage

The object-types and relationships correspond closely to those in common usage when applied to an information processing system. The main difference involved is that in most manual documentation methods, the name INPUT is related to any object which is used by a PROCESS and likewise, an OUTPUT is related to any object which is derived by a PROCESS. In general, no effort is made to distinguish between different levels of data when INPUTS and OUTPUTS are thought of in this way.

INPUTS and OUTPUTS are the names for logical collections of data whose values may eventually appear on physical media which contain data values -- such as forms, cards, tapes, messages, reports. Each individual input or output document is usually

one of a number of instances. The INPUTS and OUTPUTS being described in URL may have multiple instances. In URL the emphasis is on the logical definition rather than the physical and hence, the media or the physical format need not be specified.

The use of RECEIVES implies that some physical process will be required to receive or accept the input "document." Similarly, GENERATES implies a process will be required in the implemented target system to physically produce the OUTPUT.

	INTERFACE	INPUT	OUTPUT	SET	PROCESS
INTERFACE		GENERATES[1]			
			RECEIVES[1]		
				RESPONSIBLE FOR	
INPUT	GENERATED[1] BY				RECEIVED BY[1]
OUTPUT	RECEIVED[1] BY				GENERATED BY[1]
SET	RESPONSIBLE-INTERFACE				UPDATED[1]
PROCESS		RECEIVES[1]			
			GENERATES[1]		
				UPDATES[1]	

Table 2.1.3
URL Statements for System INPUT/OUTPUT Flow

[1] Conditional (DEPENDING ON) and repetition (FOR EACH) clauses are allowed.

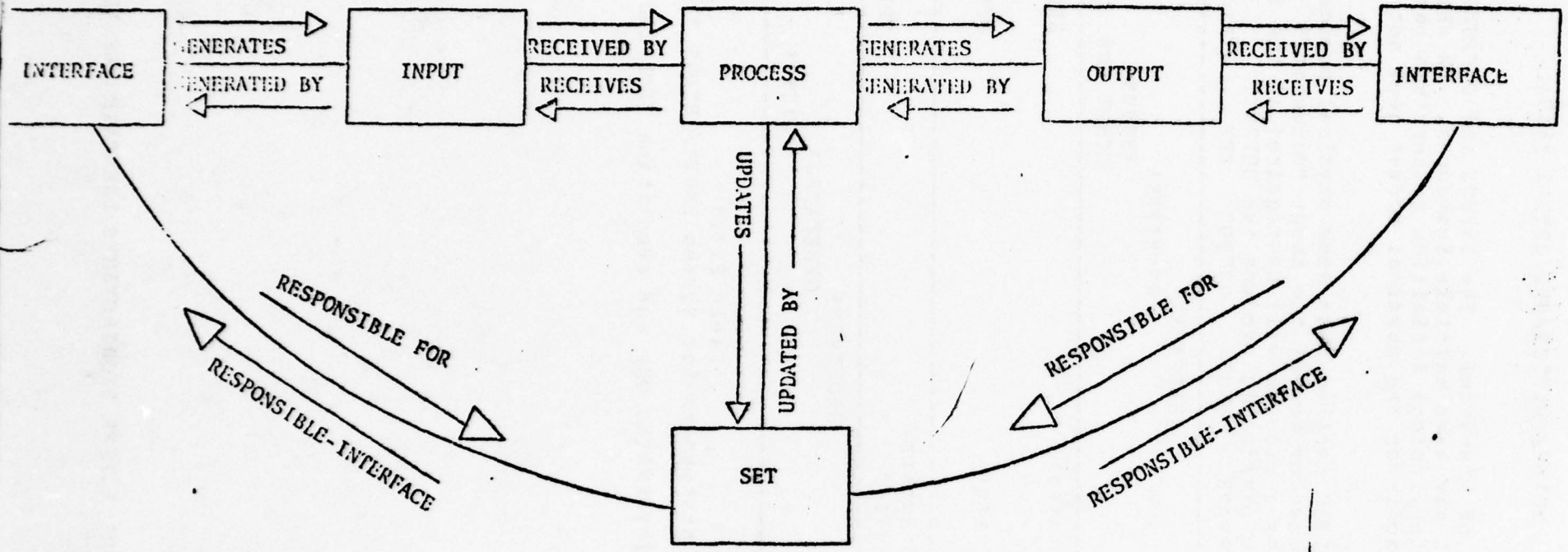


Figure 2.1.3

SYSTEM INPUT/OUTPUT FLOW

A SET can be interpreted as a master-file or data base, or more broadly to include very volatile master files such as, for example, open-order files. UPDATE implies an operation in which some data values in the SET are changed. The RESPONSIBLE FOR statement carries the common connotation of a data base "belonging" to some unit in the organization.

A REAL-WORLD-ENTITY (or INTERFACE) is an object not part of the system being described, but interacts with the system in some way. Examples are: employees, departments, companies, etc.

2.1.5 System Flow Outputs

The PICTURE report (with FLOW option in effect) can be used to present the system flow relationships (RECEIVES, GENERATES, etc.) among INPUTS, OUTPUTS, INTERFACES and PROCESSES in a graphical format.

The PROCESS-INPUT/OUTPUT report presents the same information as described above for PROCESS names, but in an alternate format. This report will also present any DESCRIPTION statements related to the PROCESS names.

2.1.6 System Flow Completeness Checks

The completeness checks that can be made for system flow completeness are:

Every INTERFACE should either (i) GENERATE some INPUT or (ii) RECEIVE some OUTPUT or (iii) be RESPONSIBLE for some SET.

Every INPUT should be GENERATED by at least one INTERFACE.

Every OUTPUT should be RECEIVED by at least one INTERFACE.

Every INPUT should be RECEIVED by at least one PROCESS.

Every OUTPUT should be GENERATED by at least one PROCESS.

The last four checks can be made by using the DATA PROCESS report.

2.2 System Structure

Definition of Structure

A number of the objects in the description of systems are related to each other by one object being a "component" of one or more other objects or "belonging" to it in some way.

Reasons for Structure

Structural relationships may be defined for one of two reasons. Structural relationships are said to arise from the "real world" if they are part of the description of the target system and its associated objects, i.e., if they really exist. Structural relationships are said to be "definitional" if they are made for convenience in the process of describing the target system but do not exist for other reasons. Real world structure must be maintained as part of the system description but definitional relationships may be discarded when no longer needed.

The description of structure permits "summarization" of the Problem Statement at various levels of the structure and, therefore, facilitates top-down or bottom-up problem definition and approval at various levels of completion.

Representation of Structure

Structural relationships are usually called trees or directed networks and represented as shown in Figure 2.2. The objects are represented by dots called nodes and the (structure) relationships by the lines, called arcs, connecting them. Trees and networks are "directed" in that the nodes are identified by the level. For example, A is a higher node than B, C or H. A node may have immediate successors or lower nodes, e.g., the immediate successors to J are E, F and G. Similarly, a node may have immediate predecessors or higher nodes, e.g., Q has immediate predecessors N and P.

Types of Structures

A node which has no predecessors, i.e., the highest node is called the root of the structure, e.g., A and M.

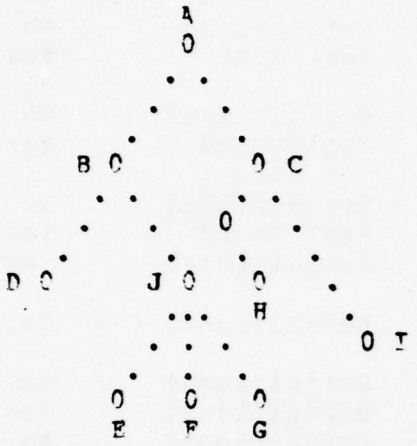
- A tree or hierarchical structure is one in which each node except the highest node has one and only one immediate predecessor (Figure 2.2a).

- A directed network structure is one in which each node except the highest node may have more than one immediate predecessor. If the structure contains no cycles, it is said to be acyclic (Figure 2.2b).

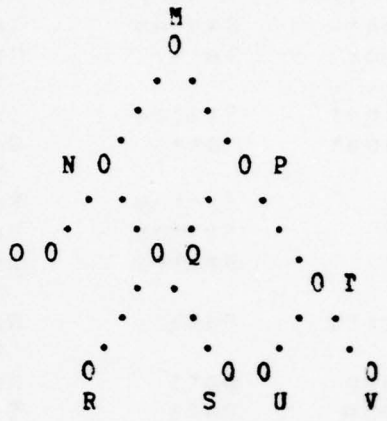
A node which has no successors is called a leaf or a terminal node.

In some cases, a structure may contain objects of different types. A structure containing objects of only one type is a "homogeneous" structure; one containing more than one type is called "non-homogeneous."

A terminal node may be assigned to the highest possible level or the lowest possible level, e.g., node D may be regarded as being on the same level as J or on the same level as E, F and G.



(a)
Tree Structure



(b)
Directed Network Structure

Figure 2.2
Tree and Network Structures

Structure in UPL

In UPL, nodes represent objects and arcs represent structural (and other) relationships. Two major types of structural relationships are available. Data structure relationships involve objects which are data elements or combinations of data elements. All other structure relationships are called system structure statements. System structure statements are described in this section, data structure statements in Section 2.3.

Table 2.2 shows possible nodes, source of relationship, type of structure, lowest unit and level of lowest unit for each type of object.

Node Object	System or Data Structure	Possible Lower Nodes	Source of Relationship	System Parameter Involved
Interface	System	Interface	Real World	No
Input	System	Inputs	Definitional	No
Input	Data	Groups/ Elements	Real World	Yes
Output	System	Outputs	Definitional	No
Output	Data	Groups/ Elements	Real World	Yes
Set	System	Sets	Definitional	No
Set	System	Entities	Real World	Yes
Set	System	Inputs/ Outputs	Definitional	Yes
Entity	Data	Groups/ Elements	Definitional	Yes
Group	Data	Groups	Definitional	Yes
Group	Data	Elements	Definitional	Yes
Process	System	Process	Definitional	No
Process	System	Process	Real World	No
Processor	System	Processor	Definitional	No
Interval	System	Intervals	Definitional	Yes

Node Object	Type of Structure[1]	Lowest Unit	Statements
Interface	Tree[2]	Interface	SUBPARTS/PART
Input	Tree[2]	Input	SUBPARTS/PART
Input	Network	Element	CONSISTS/CONTAINED
Output	Tree[2]	Output	SUBPARTS/PART
Output	Network	Element	CONSISTS/CONTAINED
Set	Network	Set	SUBSETS/SUBSET
Set	Network	Entities	CONSISTS/CONTAINED
Set	Network	Inputs/ Outputs	CONSISTS/CONTAINED
Entity	Network	Groups/ Elements	CONSISTS/CONTAINED
Group	Network	Element	CONSISTS/CONTAINED
Group	Network	Element	CONSISTS/CONTAINED
Process	Tree	Process	SUBPARTS/PART
Process	Network	Process	UTILIZES/UTILIZED
Processor	Tree	Processor	SUBPARTS/PART
Interval	Network	Interval	CONSISTS/CONTAINED

Table 2.2
CLASSIFICATION OF STRUCTURE IN URL

---

[1] A collection of trees, i.e., arborescence, is permitted
[2] Acyclic networks

2.2.1 System Structure Objects

The following types of objects may belong to system structures:

INTERFACE
INPUT
OUTPUT
PROCESS
PROCESSOR
SET

2.2.2 System Structure Relationships

SUBPARTS ARE/PART OF

These statements may be used with:

INTERFACES
INPUTS
OUTPUTS
PROCESSES
PROCESSORS

E.g., an INPUT may have SUBPARTS which are INPUTS or it may be PART OF some other INPUT.

SUBSET OF/SUBSETS ARE

A SET may be a SUBSET of some other SET or it may have other SETS as SUBSETS.

UTILIZES/UTILIZED BY

A PROCESS may UTILIZE another PROCESS or it may be UTILIZED by other PROCESSES.

2.2.3 System Structure Syntax and Semantics

The objects and relationships involved in describing system structure are shown pictorially in Figure 2.2.3 and in tabular form in Table 2.2.3.

A structure defined by the SUBPARTS/PART OF statement is a homogeneous, hierarchical tree, i.e., all nodes in a structure must be of the same type and any object can be PART OF only one immediate higher node. A node can have any number of SUBPARTS of the same type.

The relationship in a SUBSET OF/SUBSETS ARE must be homogeneous, i.e., only SETS may be SUBSETS of other SETS. The structure may

be a network, i.e., a SET can be a SUBSET of any number of other sets.

The relationship in UTILIZED BY/UTILIZES must be homogeneous, i.e., only PROCESSES can be UTILIZED by other PROCESSES. The structure may be a netowrk since a PROCESS can be UTILIZED BY any number of PROCESSES.

In general, "subdividing" an object through a structure statement does not directly imply that relationships, of other types, which held for the object also hold for its SUBPARTS. For example, suppose the problem statement has been defined:

```
INPUT              a-input;
   GENERATED BY    a-rwe;
   RECEIVED BY     a-process;
```

Then new statements are added:

```
INPUT              a-input;
   SUBPARTS        ab-input,ac-input;
```

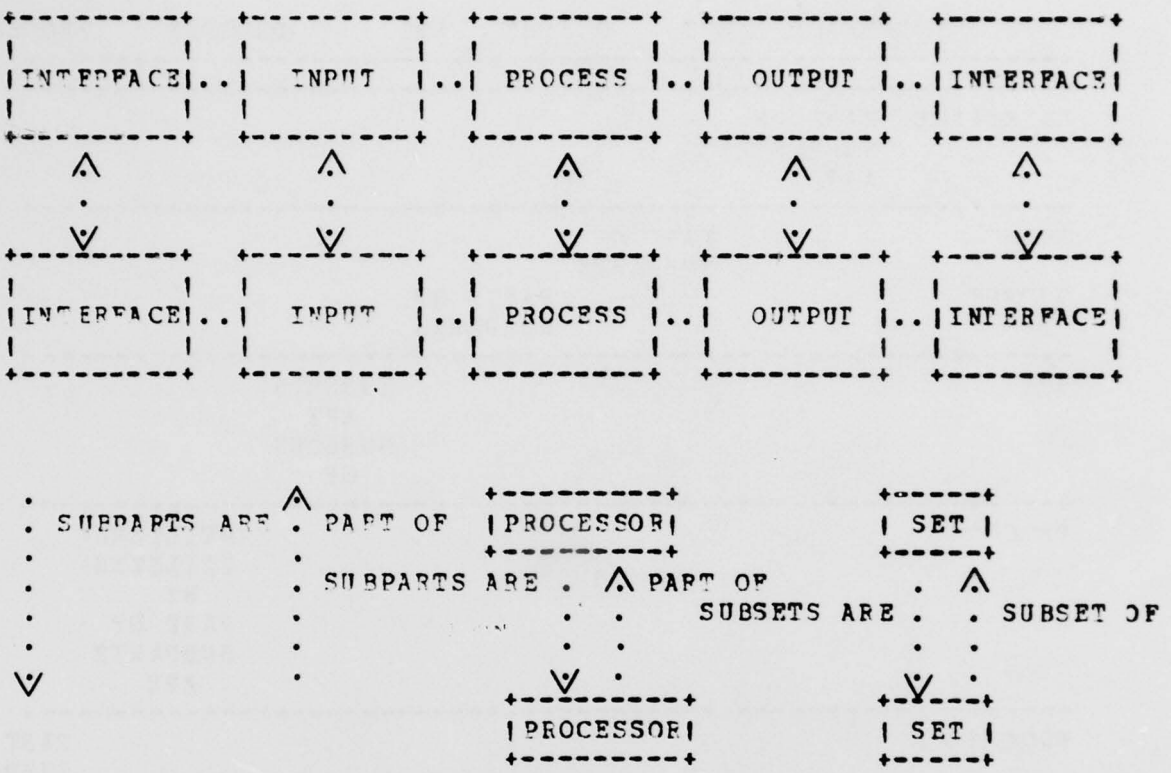


Figure 2.2.3
SOME STRUCTURAL RELATIONSHIPS EXPRESSIBLE IN URL

	INTERFACE	INPUT	OUTPUT	SET	PROCESS	PROCESSOR
INTERFACE	PART OF SUBPARTS ARE					
INPUT		PART OF SUBPARTS				
OUTPUT			PARTS OF SUBPARTS			
SET				SUBSETS ARE SUBSETS OF		
PROCESS					UTILIZED[1] UTILIZES[1] BY PART OF SUBPARTS ARE	
PROCESSOR						PART OF SUBPARTS ARE

Table 2.2.3
URL Statements for System Structure

[1] Conditional (DEPENDING ON) and repetition (FOR EACH) clauses are allowed.

The Analyzer will not automatically assume that ab-input and ac-input are GENERATED-BY a-rwe and RECEIVED by a-process. If the analyst wishes to make this statement, he should add this information explicitly:

```
INPUT              ab-input,ac-input;
   GENERATED BY    a-rwe;
   RECEIVED BY     a-process;
```

In practice if the problem had been defined from the top-down, the analyst should also have subdivided the INTERFACE and the PROCESS when the input was subdivided.

2.2.4 System Structure Common Equivalents and Usage

The tree structure of INTERFACES corresponds to the hierarchical structure of most organizations. The tree structure of INPUTS and OUTPUTS is used for convenience in definition.

It may also be used to describe:

a) A form with many copies, e.g.,

```
INPUT: FORM-A;
 SUBP: FORM-A-COPY1,
       FORM-A-COPY2;
```

or

b) Document that is generated and goes to different places, e.g.,

```
OUTPUT: FORM-A;
  SUBP: FORM-A-DEPT-X,    (names chosen according
        FORM-A-DEPT-Y,     to purpose of carrier
        FORM-A-DEPT-Z;     or final destination)
```

A PROCESS has two types of structures. The one developed by using SUBPARTS/PART OF may be used for top-down definition of the system. It may also be used to represent the structure of modules in a program (e.g., BLOCKS and PROCEDURES in a PL/1 program). In both cases, a tree structure is appropriate.

The structure of PROCESSES developed using the UTILIZED/UTILIZES may be used to represent "calls" to program or a PROCESS which is used (i.e., called from) in a number of processing sequences.

2.2.5 System Structure Reports

The FORMATTED PROBLEM STATEMENT shows the immediate structure in which an object is involved, i.e., all those objects of which it is PART OF, SUBSET OF or UTILIZED BY and those that are its SUBPARTS, SUBSETS or it UTILIZES.

The PICTURE report (with the STRUCTURE option in effect) presents the SUBPARTS/PART OF relationships for INPUTS, OUTPUTS, INTERFACES and PROCESSES in a graphical format of the immediate structure of a particular object.

The STRUCTURE report presents the same information but in a list format which displays all levels in the system structure. (The reports listed above only presents the structure levels directly above and directly below the designated object.) Loops in the system structure are detected by this report.

The STRUCTURE report presents only PART OF/SUBPARTS relationships. UTILIZES/UTILIZED BY and SUBSET OF/SUBSETS OF is only shown in the FORMATTED PROBLEM STATEMENT.

2.2.6 System Structure Completeness Checks

All the completeness statements in System Flow apply to each SUBPART as it is defined.

At each subdivision, the totality of statements about the SUBPARTS must be consistent with the statement about the objects to which the PARTS belong.

A structure of INTERFACES, since it represents the real world, cannot be checked for completeness, i.e., whether the lowest level nodes have been defined, unless terminal nodes are designated by an appropriate ATTRIBUTE.

A structure involving INPUTS/OUTPUTS is not homogeneous since the lower nodes represent GROUP or ELEMENTS. The following rules should be observed:

1) All INPUT structures having SUBPARTS should terminate in INPUTS which have a media ATTRIBUTE (whose value can be "TO BE DETERMINED," TBD) and which contain data values.

2) An INPUT should not have both a SUBPART statement and CONTAINS statement. Only the lowest level INPUT should CONTAIN ELEMENTS.

3) All OUTPUT structures having SUBPARTS should terminate in OUTPUTS which have a media ATTRIBUTE (whose value can be "TO BE DETERMINED," TBD) and which contain data values.

4) An OUTPUT should not have both a SUBPART statement and a CONTAINS statement. Only the lowest level OUTPUT should CONTAIN ELEMENTS.

When a PROCESS structure is defined using PARTS OF/SUBPARTS ARE each PROCESS may contain SUBPARTS as well as some PROCEDURE statement. A PROCESS which the analyst does not wish to subdivide further should be designated a terminal PROCESS by the use of an ATTRIBUTE statement.

A PROCESS which does not have any SUBPARTS, should have a PROCEDURE statement.

2.3 Data Structure

As was described in Section 2.2, various structural relationships can be specified in URL to relate "components" of the system. When the structual relationships being specified are applicable to data objects, the structures presented are termed "data structures."

2.3.1 Data Structure Objects

2.3.1.1 Data Definition

The basic objects for defining data are ELEMENTS and GROUPS.

ELEMENT

An ELEMENT is the basic unit of information and, therefore, cannot be subdivided. An ELEMENT is used to define an object which may take on a value. For example, if "employee information" was defined to be an ENTITY, it would not, in itself, have a value. The ELEMENTS making up "employee information" such as "age," "sex," "salary," etc., might have values for a particular instance of "employee information."

GROUP

A GROUP is used to define a collection of ELEMENTS and/or other GROUPS. This is done so that the information names can be thought to be related within the larger collection of information (INPUTS, OUTPUTS or ENTITIES). The name of the GROUP can be thought to be synonymous with the names of the GROUP's components. In the example of "employee information," the "name" of the employee may be defined as a GROUP where the constituents of the GROUP, "first name," "middle initial," "surname" may be defined as ELEMENTS. The use of GROUPS is primarily a notational convenience.

2.3.1.2 Definition of Collections of Data Values

The definition of an ELEMENT or a GROUP is like a definition of a word in a dictionary. The definition specifies how a word is to be used but does not give the instances of its use in books, paragraphs, sentences, etc.

In describing information systems, it is necessary to have types of objects which can represent the things in which, or on which, instances (values) of ELEMENTS appear. In URL, there are three such types of objects: INPUTS, OUTPUTS and ENTITIES. The difference among these types of collections is related to their role in the target system.

INPUTS

An INPUT is a collection of information produced external to the target system, but used by the target system. For example, in an inventory system, a customer order may be considered an INPUT to the system.

OUTPUTS

An OUTPUT is a collection of information produced by the target system, but which is used external to the system. For example, the paychecks produced by a payroll processing system could be thought of as OUTPUTS as they are eventually used by employees outside of the system. Once the collection has left the system, no further reference may be made to it.

ENTITIES

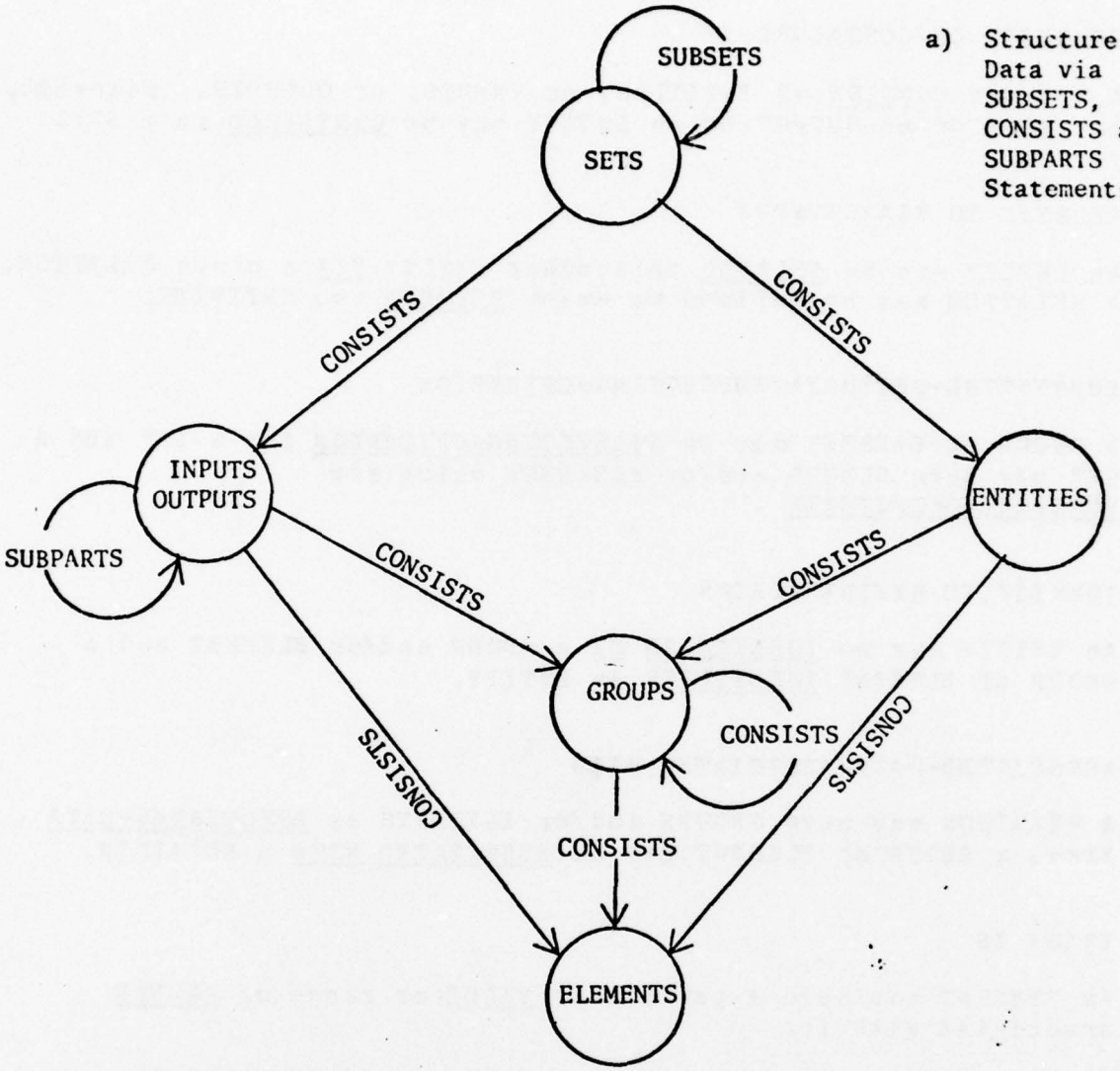
An ENTITY is a collection of information which is maintained internal to the system. ENTITIES are initially "produced" and then "maintained" using information from INPUTS and then OUTPUTS are produced using information from ENTITIES (and other sources). The "employee information" mentioned above would be defined as an ENTITY.

2.3.1.3 Relationships Among Collections of Information

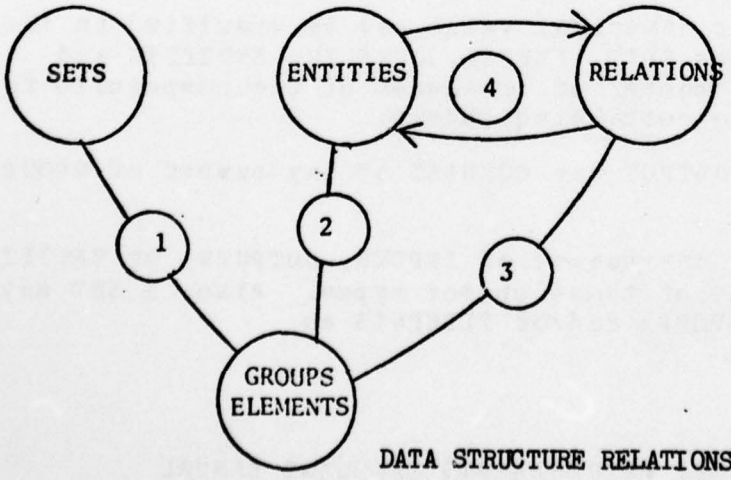
Collections of information maintained internal to the system (ENTITIES) are often "related" to each other in that there is information which is not inherent to either, but associated with both. In the example of a warehouse stock control system, information about inventory items may be related to information about their suppliers, etc. RELATIONS are used to describe these logical connections among ENTITIES.

2.3.2 Data Structure Relationships

The relationships that can be specified for data structure are shown in tabluar forms in Table 2.3.2 and Table 2.3.2.1. This information is also presented in Figure 2.3.2 in a graphical format.



a) Structure of Data via SUBSETS, CONSISTS and SUBPARTS Statements



b) Relational Structure Statements

1. SUBSETTING-CRITERIA
2. IDENTIFIED
3. ASSOCIATED-DATA
4. BETWEEN

DATA STRUCTURE RELATIONSHIPS

Figure 2.3.2

CONSISTS OF/CONTAINED IN

A SET may CONSIST of ENTITIES, or INPUTS, or OUTPUTS. Likewise, an INPUT or an OUTPUT or an ENTITY may be CONTAINED in a SET.

RELATED TO VIA/BETWEEN

An ENTITY may be RELATED to another ENTITY VIA a given RELATION. A RELATION may be defined to exist BETWEEN two ENTITIES.

SUBSETTING-CRITERIA/SUBSETTING-CRITERION

A GROUP or ELEMENT may be SUBSETTING-CRITERION for a SET and a SET may have GROUPS and/or ELEMENTS which are SUBSETTING-CRITERIA .

IDENTIFIED BY/IDENTIFIES

An ENTITY may be IDENTIFIED BY a GROUP and/or ELEMENT and a GROUP or ELEMENT IDENTIFIES an ENTITY.

ASSOCIATED-DATA/ASSOCIATED WITH

A RELATION may have GROUPS and/or ELEMENTS as ASSOCIATED-DATA . Also, a GROUP or ELEMENT may be ASSOCIATED WITH a RELATION.

VALUE IS

An ELEMENT may have a particular VALUE or range of VALUES associated with it.

CLASSIFICATION - A data object may have a CLASSIFICATION.

2.3.3 Data Structure Syntax and Semantics

A SYSTEM-PARAMETER or numerical value may be specified in the CONSISTS statement for SETS, INPUTS, OUTPUTS, ENTITIES and GROUPS to denote the number of instances of the components for a given instance of the containing object.

An ENTITY, INPUT or OUTPUT may CONSIST of any number of GROUPS and/or ELEMENTS.

A SET may CONSIST of any number of INPUTS, OUTPUTS, or ENTITIES, but not a combination of these object types. Also, a SET may have any number of GROUPS and/or ELEMENTS as SUBSETTING-CRITERIA.

An ENTITY can be IDENTIFIED by any number of GROUPS and/or ELEMENTS, and be RELATED to any number of ENTITIES. However, for each unique pair of ENTITIES, a unique RELATION must be defined. E.g., if a RELATION between E1 and E2 is defined as R1, a RELATION between E1 and E3 cannot also be called R1.

A RELATION may only be defined to be BETWEEN a single pair of ENTITIES. A different RELATION must be defined for each ENTITY pair. A RELATION may have any number of GROUPS and/or ELEMENTS as ASSOCIATED-DATA.

A GROUP may be CONTAINED in any number of GROUPS, ENTITIES, INPUTS and/or OUTPUTS. A GROUP may also IDENTIFY any number of ENTITIES, be SUBSETTING-CRITERION for any number of SETS and be ASSOCIATED WITH any number of RELATIONS. In addition, a GROUP may CONSIST of any number of GROUPS and/or ELEMENTS.

	INPUTS	OUTPUTS	SET	ENTITY	GROUPS	ELEMENTS
INPUTS			CONTAINED IN		CONSISTS OF	CONSISTS OF
OUTPUTS			CONTAINED IN		CONSISTS OF	CONSISTS OF
SET	CONSISTS OF	CONSISTS OF		CONSISTS OF		
ENTITIES			CONTAINED IN		CONSISTS OF	CONSISTS OF
GROUPS	CONTAINED IN	CONTAINED IN		CONTAINED IN	CONSISTS OF CONTAINED IN	CONSISTS OF
ELEMENTS	CONTAINED IN	CONTAINED IN		CONTAINED IN	CONTAINED IN	

Table 2.3.2
URL Statements for Data STRUCTURE Relationships

	SET	ENTITY	RELATION	GROUPS	ELEMENTS
SET				SUBSETTING-CRITERIA	SUBSETTING CRITERIA
ENTITIES		RELATED TO	R/VIA	IDENTIFIED BY	IDENTIFIED BY
RELATION		BETWEEN		ASSOCIATED-DATA	ASSOCIATED DATA
GROUPS	SUBSETTING-CRITERION	IDENTIFIES	ASSOCIATED WITH		
ELEMENT	SUBSETTING-CRITERION	IDENTIFIES	ASSOCIATED WITH		VALUES

Table 2.3.2.1
URL Definitional Statements Relating
SETS, ENTITIES, RELATIONS, GROUPS and ELEMENTS

An ELEMENT may be CONTAINED in any number of GROUPS, ENTITIES, INPUTS, and/or OUTPUTS. An ELEMENT may also be used to IDENTIFY any number of ENTITIES, be SUBSETTING-CRITERION for any number of SETS, and be ASSOCIATED WITH any number of RELATIONS. In addition, an ELEMENT may take on a particular numerical VALUE or a range of VALUES.

2.3.4 Data Structure Common Equivalents and Usage

The names URL uses to define data structures are very close to most terminology in this field. For example, ELEMENTS are often referred to as "items," "data items," or "fields" in other data structure terminologies. GROUPS are sometimes referred to as "segments" or "data aggregates." ENTITIES are sometimes called "records" and SETS sometimes "files" or "data bases."

If a SET is intended to represent a "file" where ENTITIES are "records," the following options are available in describing the file structure.

a) If the SET CONSISTS of only one type of ENTITY, then:

- ENTITY occurrences within the SET may be ordered and so a RELATION to represent this ordering may be defined.[1]

- ENTITY occurrences within the SET may not be ordered. A

---

[1] If more than one RELATION is to be defined for ENTITIES within a given SET, a SET (which is a SUBSET of the given SET) should be defined for each RELATION.

RELATION to represent this need not be defined.

- ENTITY occurrences may be RELATED to each other based on some criteria. A RELATION should be defined to describe this relationship.

b) If the SET CONSISTS of more than one type of ENTITY:

- ENTITY occurrences may be ordered. A RELATION should be defined for each ordering. [1]

- ENTITY occurrences may not be ordered.

- ENTITY occurrences may be RELATED to other ENTITY types (for each other). A RELATION should be defined to describe each of these relationships.

The IDENTIFIES statement for GROUPS or ELEMENTS may be used to define keys. It is meant that the designated GROUP or ELEMENT may be used when searching for a particular ENTITY (record) occurrence.

HOW TO USE THE RELATION SECTION TO EXPRESS LOGICAL CONNECTION IN PROBLEM STATEMENT

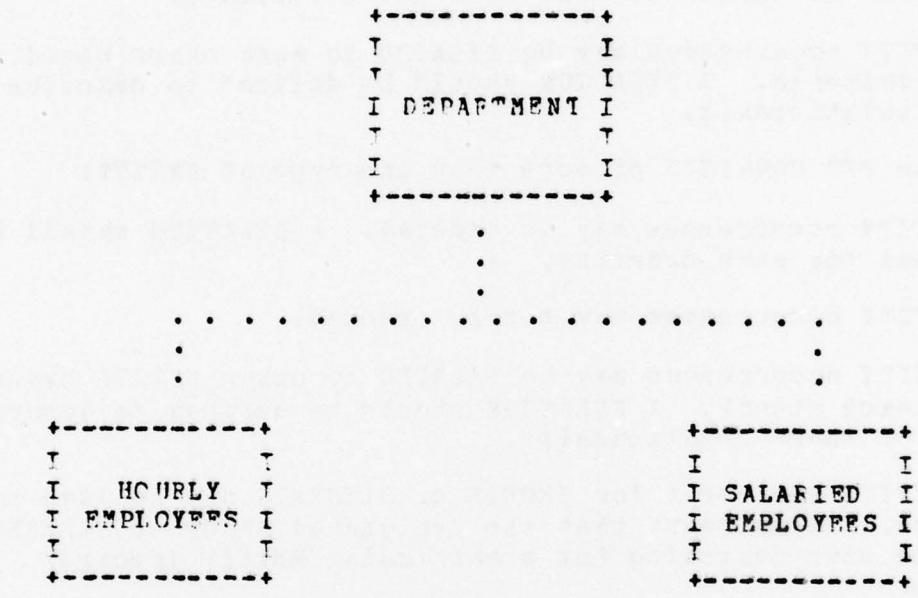
Step 1

a) Determine symbolic (URL) name for the RELATION. It is recommended that the name denotes the type of connection that it will supply.

b) Determine which ENTITIES the RELATION connects and the direction of the connection. Use the URL BETWEEN and CONNECTIVITY statements to state this information.

Example:

Suppose the analyst has the following (logical) view of his data:



The URL statements that define the two RELATIONS are:

```
RELATION dept-to-hourly-employees;
BETWEEN dept AND hourly-employees;
CONNECTIVITY IS 1 TO max-dept-hourly-employment;

RELATION dept-to-salaried-employees;
BETWEEN dept AND salaried-employees;
CONNECTIVITY IS 1 TO max-dept-salaried-employment;
```

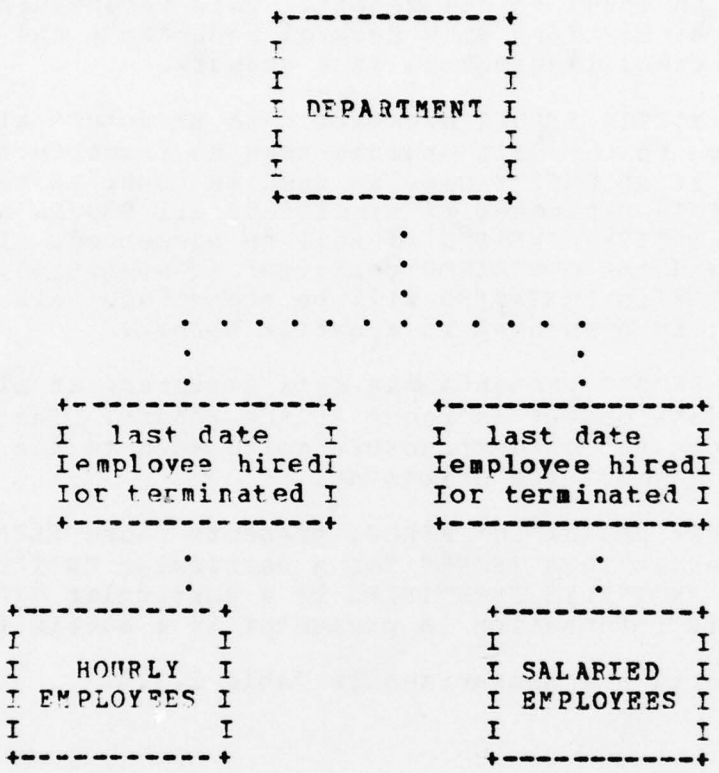
Step 2

a) Determine if any data has been defined to be CONTAINED in both ENTITIES. Analyze this data and determine ENTITIES or ASSOCIATED DATA statements.

b) Determine if any additional data is needed to describe the RELATION and, if so, this data should be defined as ASSOCIATED DATA.

Example:

A more refined and detailed logical view of the data given above might be:



Step 3

a) Determine the RELATION's CARDINALITY

b) Determine the PROCESSES that utilize the RELATION and those PROCESSES that add, delete or modify the connection occurrences of the RELATION.

Results:

The analyst has information that is required for physical design. There is a connection between the programming requirement and the data base. The data base may have to be revised to be receptive to the processing restrictions.

For an example, see RELATION Definition Form.

2.3.5 Data Structure Outputs

The CONSISTS COMPARISON REPORT presents the lowest level data objects (usually ELEMENTS) in the data structure of the data objects used as input to the report. This information is presented in matrix form with several redundancy and completeness check diagnostics in a summary.

The CONSISTS MATRIX REPORT presents data structure at a given level relative to the data objects used as input to the report. For example, if an ENTITY name is used as input to the report and the CONSISTS parameter is specified, all GROUPS and/or ELEMENTS the ENTITY CONSISTS of will be presented. If the ENTITY name and the CONTAINED parameter is specified, all those SETS the ENTITY is CONTAINED will be presented. All information in the report is presented in a matrix format.

The CONTENTS REPORT presents the data structure at all levels for a given data object as input to the report. The CONTENTS REPORT presents the data structure going down to the lowest specified in the problem statement.

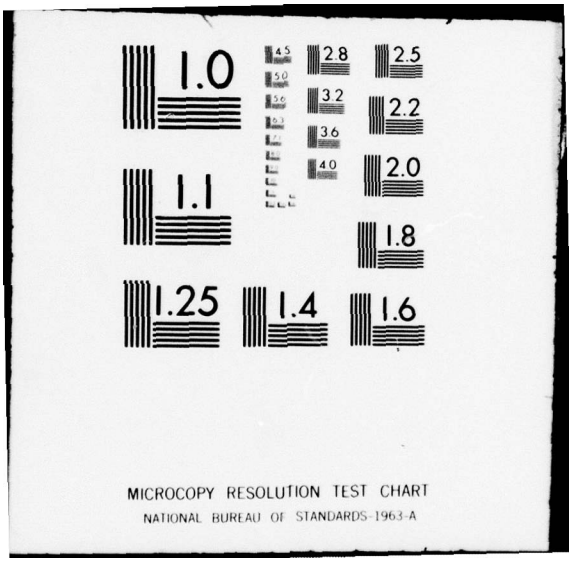
The IDENTIFIER INFORMATION REPORT presents those ELEMENTS and/or GROUPS defined as IDENTIFIERS for a particular ENTITY or presents the ENTITIES IDENTIFIED by a particular GROUP or ELEMENT. This information is presented in a matrix format.

The four reports are summarized in Table 2.3.5.

AD-A058 629 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART I. (DESCRI--ETC(U)
JUL 78 F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-130-VOL-1 NL

2 OF 3
AD
A058629


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

CONSISTS COMPARISON	CONTENTS	CONSISTS MATRIX CONSIST	CONTAINED	FPS
SET	X	X	NO	X
ENTITY	X	X	X	X
INPUT	X	X	X	X
OUTPUT	X	X	X	X
GROUP	X	X	X	X
ELEMENT			X	X

ROW ID	HIGHEST LEVEL ID	COL ID	ROW ID	OBJECT

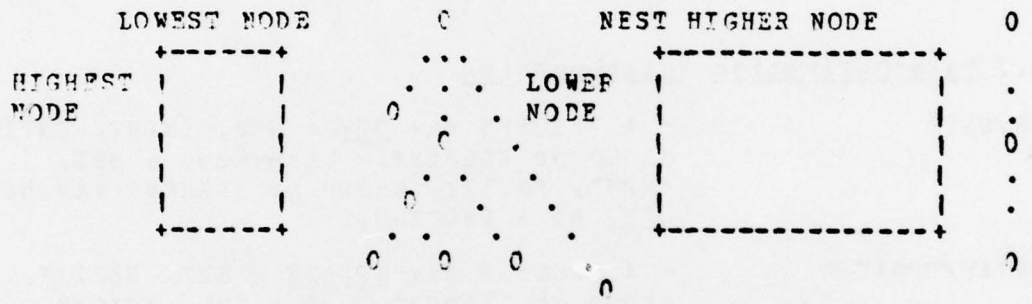


TABLE 2.3.5

2.3.6 Data Structure Completeness Checks

All SETS should "eventually " consist of INPUTS, OUTPUTS or ENTITIES.

All INPUTS at the lowest level should consist of GROUPS and ELEMENTS. Any GROUPS should be reducible to ELEMENTS.

ALL OUTPUTS at the lowest level should consist of GROUPS and ELEMENTS. Any GROUPS should be reducible to ELEMENTS.

2.4 Data Derivation

An information processing system exists to process data, i.e., to produce values of data elements, or groups of data elements, from values of other data elements or groups. This transformation is known by different names such as process, procedure, function, operation, activity, etc. In URL the term PROCESS is used.

The term "data derivation" includes the actions of USING, UPDATING and DERIVING data objects. The data objects that are

involved can be INPUTS, OUTPUTS, SETS, ENTITIES, GROUPS and ELEMENTS.

2.4.1 Data Derivation Objects

The objects involved in data derivation are:

PROCESS
SET
INPUT
OUTPUT
ENTITY
GROUP
ELEMENT
RELATION

2.4.2 Data Derivation Relationships

USES/USED	- A PROCESS may USE a SET, INPUT, ENTITY, GROUP or ELEMENT. Likewise, a SET, INPUT, ENTITY, GROUP or ELEMENT may be USED by a PROCESS.
UPDATES/UPDATED	- A PROCESS may UPDATE a SET, ENTITY, GROUP or ELEMENT, and a SET, ENTITY, GROUP or ELEMENT may be UPDATED by a PROCESS.
DERIVES/DERIVED	- A PROCESS may DERIVE a SET, OUTPUT, ENTITY, GROUP or ELEMENT, and a SET, OUTPUT, ENTITY, GROUP or ELEMENT may be DERIVED by a PROCESS.
MAINTAINS/MAINTAINED	- A PROCESS may MAINTAIN a RELATION, and a RELATION may be MAINTAINED by a PROCESS.
PROCEDURE	- A PROCESS may have a PROCEDURE associated with it. The PROCEDURE is a comment entry and may consist of any text.
DERIVATION	- A RELATION or SET may have a DERIVATION associated with it in the form of a comment entry.

2.4.3 Data Derivation Syntax and Semantics

The objects and relationships involved in describing "data derivation" are shown pictorially in Figure 2.4.3 and in tabular form in Table 2.4.3. Table 2.4.3.1 shows how the different types of objects can appear in the data derivation statements. Table 2.4.3.2 contrasts the syntax and semantics of the System Flow Statements (RECEIVES and GENERATES) with that of the data derivation statements.

Whenever INPUT, OUTPUT, ENTITY or SET are used in a data derivation statement, these objects are interpreted to mean the data values contained in them.

A PROCESS may USE any number of INPUTS, SETS, ENTITIES, GROUPS and ELEMENTS. An optional UPDATE or DERIVE clause can be used in conjunction with the USE statement in the following manner:

USES E1 TO DERIVE E2:

Where E2 is any number of data objects that can be DERIVED by a PROCESS.

A PROCESS can UPDATE any number of SETS, ENTITIES, GROUPS and ELEMENTS. An optional USING clause can be used in conjunction with the UPDATE statement in the following manner:

UPDATES E1 USING E2;

Where E2 is any number of data objects that can be USED by a PROCESS.

A PROCESS can DERIVE any number of OUTPUTS, SETS, ENTITIES, GROUPS and ELEMENTS. An optional USING clause can be used in conjunction with the DERIVE statement in the following manner:

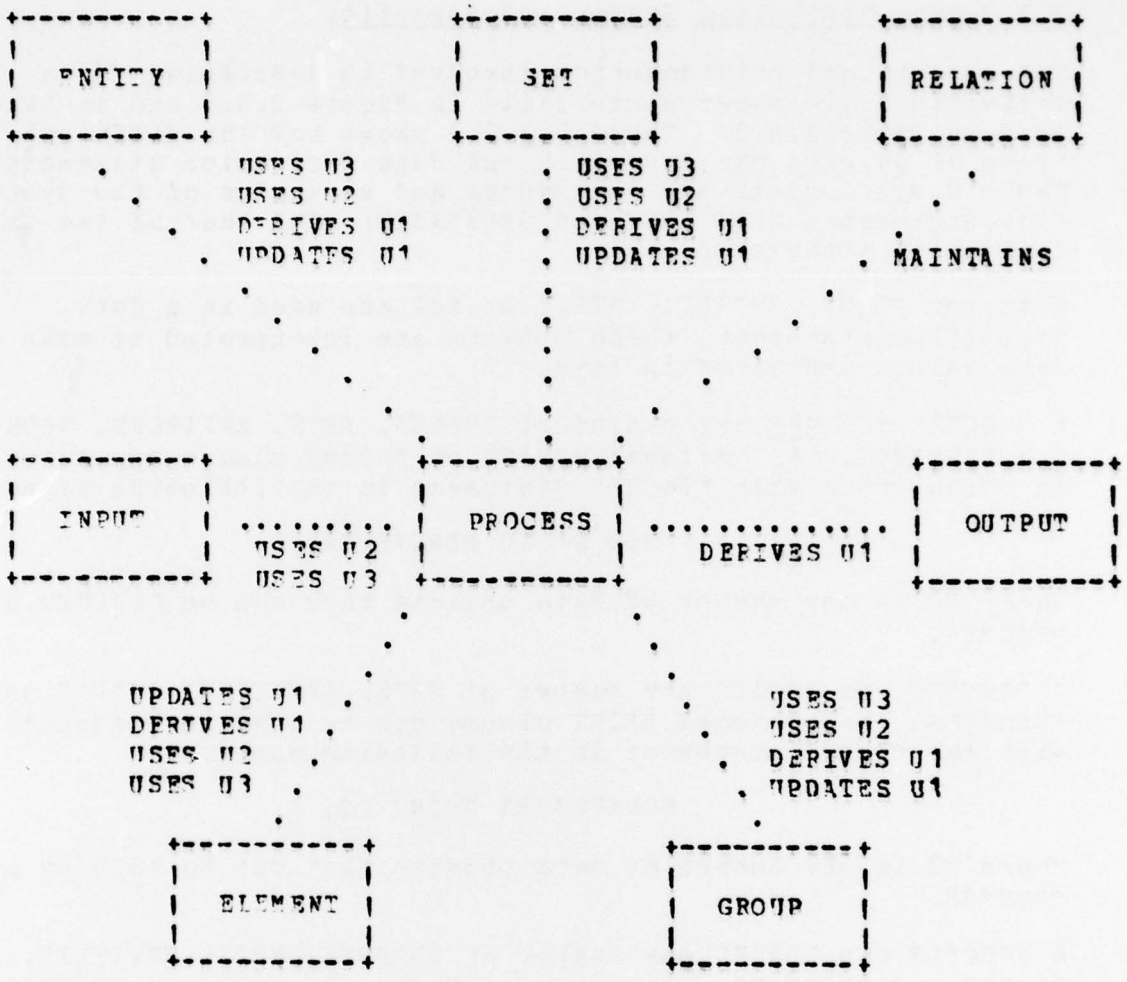
DERIVES E1 USING E2;

Where E2 is any number of data objects that may be USED by a PROCESS.

An INPUT, SET, ENTITY, GROUP or ELEMENT can be USED by any number of PROCESSES. An optional DERIVE or UPDATE clause may be used in conjunction with the USED statement in the following manner:

USED BY P1 TO DERIVE E2;

Where E2 is any number of data objects that can be DERIVED by a PROCESS.



U1, U2 and U3 are optional
U1 using ELEMENT, GROUP, ENTITY and INPUT
U2 to derive ELEMENT, GROUP, ENTITY, SET and OUTPUT
U3 to update ELEMENT, GROUP, ENTITY and SET

Figure 2.4.3
URL STATEMENTS FOR DATA MANIPULATION

Section Type	Object Name in Statement	INPUT	OUTPUT	SET	ENTITY
INPUT			TO DERIVE[3]	TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
OUTPUT		USING[4]		USING[4]	USING[4]
SET		USING[5]			
			TO DERIVE[3]	TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
				USING[5]	USING[5]
ENTITY		USING[5]			
			TO DERIVE[3]	USING[5]	USING[5]
				TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
GROUP		USING[5]			
			TO DERIVE[3]	USING[5]	USING[5]
				TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
ELEMENT		USING[5]			
			TO DERIVE	USING[5]	USING[5]
				TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
PROCESS		USING[7]			
			DERIVES TO DERIVE[8]	DERIVES USES UPDATES USING[7] TO DERIVE/UPDATE[8]	DERIVES USES UPDATES USING[7] TO DERIVE/UPDATE[8]

	RELATION	GROUP	ELEMENT
INPUT		TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE
OUTPUT		USING[4]	USING[4]
SET		TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
		USING[5]	USING[5]
ENTITY		USING[5]	USING[5]
		TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
GROUP		USING[5]	USING[5]
		TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
ELEMENT		USING[5]	USING[5]
		TO DERIVE/UPDATE[3]	TO DERIVE/UPDATE[3]
PROCESS	MAINTAINS	DERIVES	DERIVES
		USES	USES
		UPDATES	UPDATES
		USING[7]	USING[7]
		TO DERIVE/UPDATE[8]	TO DERIVE/UPDATE[8]

(see following page for footnotes)

Table 2.4.3 UPL Statements Related to Derivation Definition

	PROCESS
INPUT	USED BY
OUTPUT	DERIVED BY
SET	USED BY
	UPDATED BY
	DERIVED BY
ENTITY	DERIVED BY
	UPDATED BY
	USED BY
RELATION	MAINTAINED BY
GROUP	DERIVED BY
	UPDATED BY
	USED BY
ELEMENT	DERIVED BY
	UPDATED BY
	USED BY
PROCESS	UTILIZES
	UTILIZED BY

Table 2.4.3
URL Statements Related to Derivation Definition
(Continued)

Footnotes:

[3] Used in conjunction with the USED BY statement.

[4] Used in conjunction with the DERIVED BY statement, may have DEPENDING ON and FOR EACH clauses.

[5] Used in conjunction with DERIVED BY and UPDATED BY statement, may have DEPENDING ON and FOR EACH clauses.

[7] Used in conjunction with DERIVES and UPDATES statement, may have DEPENDING ON and FOR EACH clauses.

[8] Used in conjunction with USES statement.

USES

	ELEMENT	GROUP	INPUT	OUTPUT	ENTITY	SET
USES	X	X	X		X	X
USES TO DERIVE	X	X	X		X	X
USES TO UPDATE	X	X	X		X	X
DERIVES						
DERIVES/{USING {DEPENDING ON} { FOR EACH }	X	X	X		X	X
UPDATES						
UPDATES/{USING {DEPENDING ON} { FOR EACH }	X	X	X		X	X

DERIVES or UPDATES

	ELEMENT	GROUP	INPUT	OUTPUT	ENTITY	SET
USES						
USES TO DERIVE	X	X		X	X	X
USES TO UPDATE	X	X			X	X
DERIVES	X	X		X	X	X
DERIVES/{USING {DEPENDING ON} { FOR EACH }	X	X		X	X	X
UPDATES	X	X			X	X
UPDATES/USING	X	X			X	X

Table 2.4.3.1
Data Derivation Relationships for
USES, UPDATES and DERIVES Statements

	ELEMENT	GROUP	INPUT
RECEIVES	Not Allowed	Not Allowed	Every INPUT should be RECEIVED by at least one PROCESS
GENERATES	Not Allowed	Not Allowed	Not Allowed
USES	Every ELEMENT should be used by at least one PROCESS	At least one ELEMENT in the GROUP is used by the PROCESS	At least one ELEMENT in the INPUT is used by the PROCESS
DERIVES	Value of an ELEMENT is derived by the PROCESS	Value of at least one ELEMENT in the GROUP is derived by the PROCESS	Not Allowed
UPDATES	1) Value of an ELEMENT is updated by the PROCESS 2) ELEMENT should be CONTAINED in at least one ENTITY	1) Value of at least one ELEMENT in the GROUP is updated by PROCESS 2) GROUP should be CONTAINED in at least one ENTITY	Not Allowed

Table 2.4.3.2
Data Derivation (PROCESS) Semantics

	OUTPUT	ENTITY	SET
RECEIVES	Not Allowed	Not Allowed	Not Allowed
GENERATES	Every OUTPUT should be GENERATED by at least one PROCESS	Not Allowed	Not Allowed
USES	Not Allowed	At least one ELEMENT in the ENTITY is used by the PROCESS	At least one ELEMENT in the SET is used by PROCESS
DERIVES	Value of at least one ELEMENT in OUTPUT is derived by the PROCESS	At least one ELEMENT in the ENTITY is derived	At least one ELEMENT in the SET is derived
UPDATES	Not Allowed	Value of at least one ELEMENT in the ENTITY is updated by the PROCESS	Value of at least one ELEMENT in the SET is updated by the PROCESS

Table 2.4.3.2
(Continued)

A SET, ENTITY, GROUP or ELEMENT may be UPDATED by any number of PROCESSES. An optional USING clause may be used in conjunction with the UPDATED statement in the following manner:

UPDATED BY P1 USING E2;

Where E2 is any number of data objects that may be USED by a PROCESS.

The optional clause DEPENDING ON may be used in conjunction with the UPDATED statement in the following manner:

UPDATED BY P1 DEPENDING ON E3

where E3 is list-of-elements or condition-names.

The optional clause FOR EACH may be also used in the following manner:

UPDATED BY P1 FOR EACH E4

Where E4 is list-of-entity, input, output, group, element, set-names.

The three optional clauses can be used together:

UPDATED by P1 USING E2 DEPENDING ON E3 FOR EACH E4

An OUTPUT, SET, ENTITY, GROUP or ELEMENT may be DERIVED by any number of PROCESSES. An optional USING clause may be used in conjunction with the DERIVED statement in the following manner:

DERIVED BY P1 USING E2;

Where E2 is any number of data objects that may be USED by a PROCESS.

The optional clause DEPENDING ON may be used in conjunction with the DERIVED statement in the following manner:

DERIVED BY P1 DEPENDING ON E3

where E3 is list-of-elements or condition-names.

The optional clause FOR EACH may be also used in the following manner:

DERIVED BY P1 FOR EACH E4

Where E4 is list-of-entity, input, output, group, element, set-names.

The three optional clauses can be used together:

DERIVED by P1 USING E2 DEPENDING ON E3 FOR EACH E4

A RELATION may be MAINTAINED by any number of PROCESSES, and a PROCESS may MAINTAIN any number of RELATIONS.

The optional clauses DEPENDING ON and FOR EACH can be used in conjunction with the MAINTAINS statement in the following manner:

MAINTAINED BY P1 DEPENDING ON E3 FOR EACH E4

Where P1, E3, E4 are the same as above.

A PROCESS may have any number of PROCEDURE comment entries specified, but all the comment entries will be combined into one PROCEDURE comment entry when presented in any URA report.

A SET or RELATION may have any number of DERIVATION comment entries specified, but all these comment entries will be combined into one DERIVATION comment entry when presented in any URA report.

When a collection of data (e.g., an ENTITY or GROUP) is USED, this implies that at least one ELEMENT within the collection (assuming the collection is, or will be, broken down to one ELEMENT level) is USED.

When a collection of data is UPDATED, this implies that at least one ELEMENT within the collection is UPDATED.

When a collection of data is DERIVED, this implies that at least one ELEMENT within the collection is DERIVED.

Whenever PROCESSES or PROCESSORS access data, whether deriving, updating or using it, the CLASSIFICATION of the data and the SECURITY-ACCESS-RIGHTS of PROCESS or PROCESSOR should match. In order to match, the PROCESS or PROCESSOR should have SECURITY-ACCESS-RIGHTS at a level greater than or equal to the CLASSIFICATION of the data object.

2.4.4 Data Derivation Common Equivalents and Usage

In most manual documentation methods, the information related to "data derivation" is usually implicitly included in flow charts. Flow charts usually contain more than just the "data derivation," and, consequently, data derivation may not be clearly presented.

A PROCESS that is UTILIZED represents some function within the system that is incorporated by two or more higher level PROCESSES. For example, a validation routine might be a PROCESS UTILIZED by several other PROCESSES to perform their defined functions.

The PROCEDURE comment entry within the PROCESS description may be used to describe the algorithms required to define the PROCESS. Since the PROCEDURE is text, decision tables may be included.

The DERIVATION comment entry within the SET or RELATION descriptions may be used to define the rules to derive an occurrence of a RELATION between two ENTITIES, or occurrences of a member within a SET.

2.4.5 Data Derivation Outputs

The PICTURE report (with the DATA option in effect) can be used to present data derivation relationships (USES, UPDATES, and DERIVES) among SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS, ELEMENTS and PROCESSES in a graphical format.

The EXTENDED PICTURE report (with the DATA-FLOW option in effect) can be used to present the all data derivation relationships (USES, UPDATED, DERIVED, GENERATED, and RECEIVED) among SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS, ELEMENTS, PROCESSES, and INTERFACES in a graphical tree-structured format looking FORWARD or BACKWARD in the tree.

The PROCESS-INPUT/OUTPUT report presents most of the information as described above for PROCESS names only, but in an alternate format. This report will also present any DESCRIPTION and PROCEDURE comment entries related to the PROCESS names.

The DATA PROCESS report presents the interaction of data objects with PROCESSES in a matrix format. This has the advantage of presenting the dependencies of data by PROCESSES for the entire system. A second matrix is also produced to present the degree in which PROCESSES interact with each other; i.e., to produce data that other PROCESSES use or to require data that other PROCESSES produce.

2.4.6 Data Derivation Completeness Checks

1) Every PROCESS should acquire some data either by USING or UPDATING.
2) Every PROCESS should produce data by DERIVING or by UPDATING.
3) Every SET should be USED or UPDATED by some PROCESS.
4) Every ENTITY should be USED or UPDATED by some PROCESS.
5) Every ELEMENT in an ENTITY should serve at least one purpose:
   - IDENTIFIER of the ENTITY
   - USED by some PROCESS, or
   - UPDATED by some PROCESS.
6) Processing statements in which GROUPS appear should apply to at least one ELEMENT in the GROUP.

7) Every ELEMENT CONTAINED in an INPUT should be USED in some way.
8) Every ELEMENT CONTAINED in an ENTITY should serve a purpose.
9) Every ELEMENT CONTAINED in an OUTPUT should be DERIVED by some PROCESS.
10) An ELEMENT CONTAINED in an INPUT should not be DERIVED.
11) An ELEMENT should only be DERIVED once.
12) Every ELEMENT USED by a PROCESS should be available from some source:
    i) INPUT
    ii) DERIVED by some other PROCESS
    iii) From an ENTITY.

2.5 System Size

The complete specification of requirements for the target system requires statement of parameters that specify the volume of work that the system will have to do and the amount of resources that it will require. Two types of data should be given.

Size - number of members in each SET, number of repetitions in each repeating GROUP in an INPUT, etc.

Volume - number of instances of INPUTS and OUTPUTS, number of times PROCESSES will be executed, etc. in a given period of time.

In URL, the parameters which characterize size are called SYSTEM-PARAMETERS; they can be name symbolically and their values expressed numerically.

2.5.1 System Size Objects

SYSTEM-PARAMETERS - an object which affects the size of the system. It is given a name and may be given a numeric value.

INTERVAL - an object representing some time period such as a week, year, millisecond, planning period, etc.

2.5.2 System Size Relationships

VALUES

A SYSTEM-PARAMETER may have a VALUE, or a range of VALUES. An ELEMENT may also have a VALUE or range of VALUES associated with it.

CARDINALITY

An ENTITY, or SET, or RELATION may have a CARDINALITY.

CONNECTIVITY

A RELATION may have a CONNECTIVITY defined by specifying two SYSTEM-PARAMETERS.

HAPPENS

An INPUT, OUTPUT, EVENT, or PROCESS may HAPPEN:

system-parameter TIMES-PER interval
EVERY system-parameter interval
WITHIN system-parameter interval AFTER event
system-parameter interval AFTER event

CONSISTS

A SET may CONSIST of a SYSTEM-PARAMETER (number) of ENTITIES, INPUTS, or OUTPUTS. An INPUT, OUTPUT, ENTITY, or GROUP may CONSIST of a SYSTEM-PARAMETER (number) of GROUPS and/or ELEMENTS. An INTERVAL may CONSIST of a SYSTEM-PARAMETER (number) of INTERVALS.

2.5.3 System Size Syntax and Semantics

The objects and relationships involved in describing system size are shown pictorially in Figures 2.5.3, 2.5.3.1 and 2.5.3.2, and in tabular form in Table 2.5.3.

The VALUE or VALUES associated with a SYSTEM-PARAMETER or ELEMENT must be numeric and once a VALUE (or VALUES) has been assigned, no other VALUES may be given to it.

CARDINALITY specifies a number of occurrences. With respect to SETS, it specifies the number of ENTITIES, INPUTS, or OUTPUTS that may be CONTAINED in the SET at any one time. With respect to ENTITIES, it specifies the number of occurrences of a particular ENTITY in the system at any one time. With respect to RELATIONS, it specifies the number of connections made between ENTITIES via a particular RELATION. A particular ENTITY, SET, or RELATION may have only one CARDINALITY.

CONNECTIVITY specifies the structure and magnitude of a RELATION. A particular RELATION may have only one CONNECTIVITY.

The HAPPENS statement specifies the number of occurrences of an INPUT, OUTPUT, EVENT, or PROCESS in a given time interval.

The CONSISTS statement used in conjunction with a SYSTEM-PARAMETER specifies that for each occurrence of a given SET, e.g., the data CONTAINED in it occurs the designated number of times. Any particular data object may only consist of another data object, one given SYSTEM-PARAMETER number of occurrences.

2.5.4 System Size Common Equivalent and Usage

In the usual methods of system documentation, description of size and volume aspects are incorporated into the descriptions of other objects as numerical values.

One important feature of URL in specifying size is that it permits, and in fact encourages, all such specifications to be symbolic, i.e., each parameter is given a name. Consequently, all situations in which a given parameter appears can be collected and examined. Numerical values need only be assigned at the time at which they are definitely needed. For example, when a system is initially being described, it may only be known that the group "job-data" CONSISTS of the element "occupation." It may not be known or not specified until much later that job-data CONSISTS of 3 or 6 occurrences of "occupation."

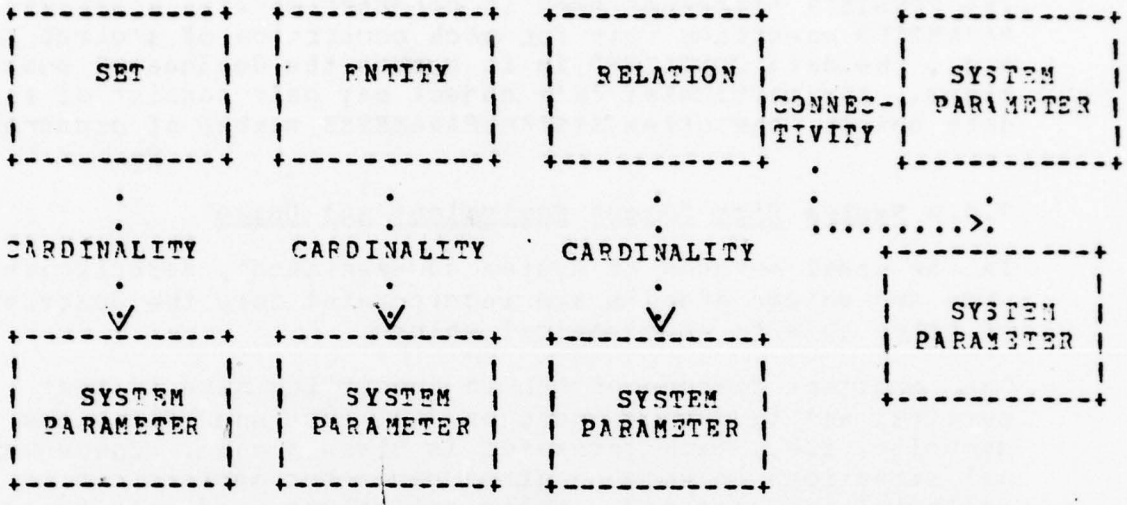


Figure 2.5.3 Relation of Objects to a SYSTEM PARAMETER

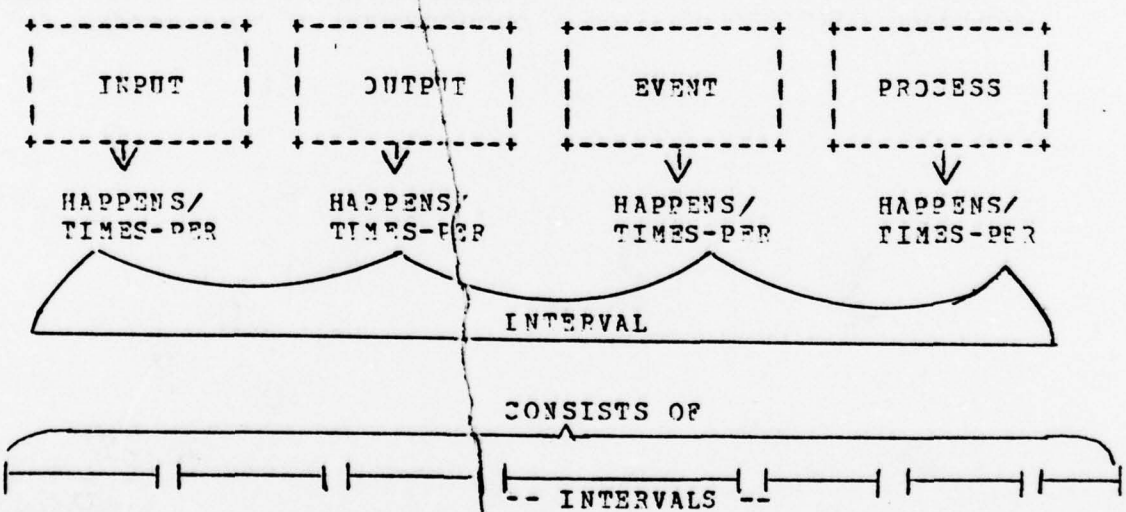


Figure 2.5.3.1 Relation of Objects to an INTERVAL

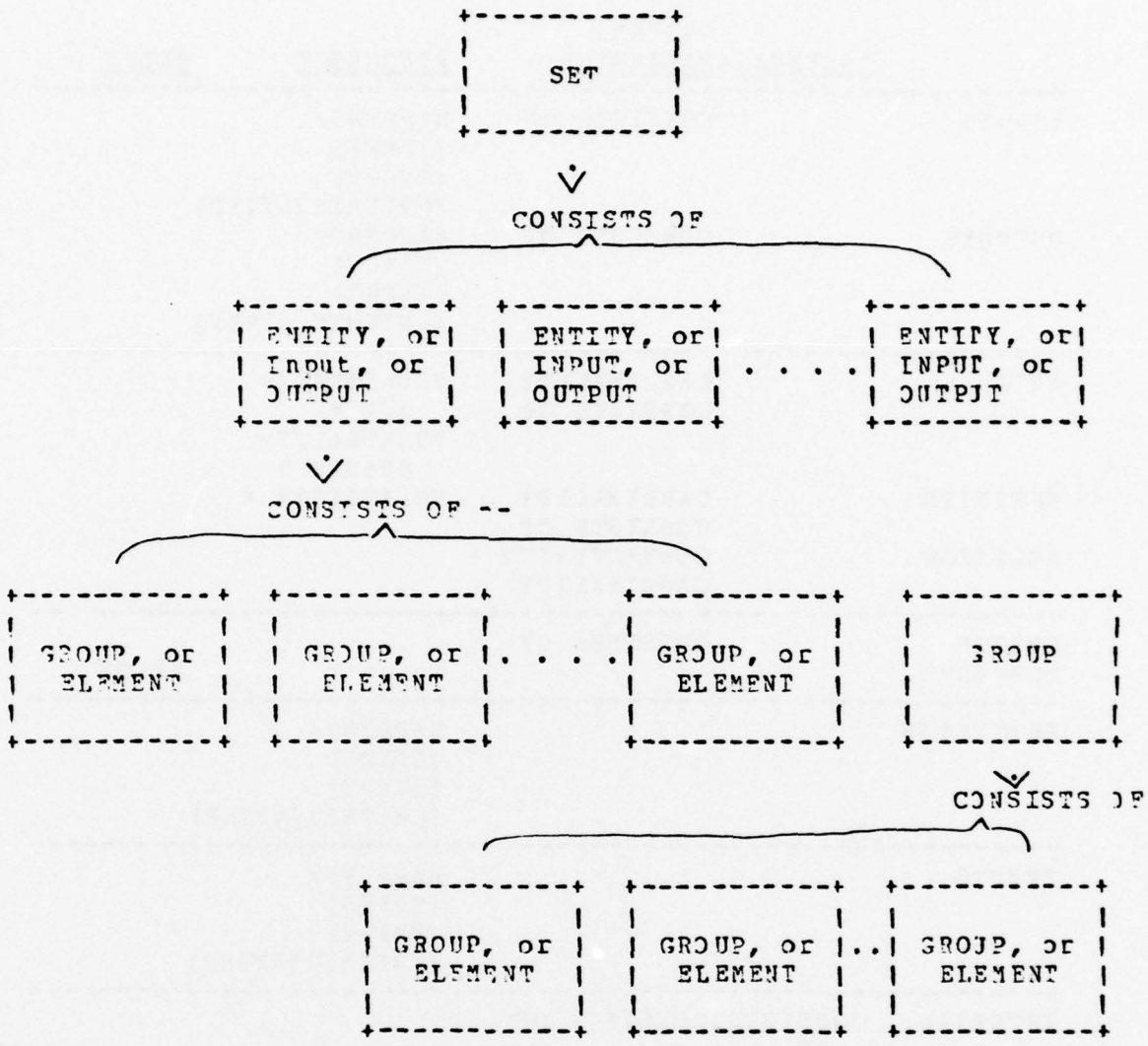


Figure 2.5.3.2 Relation of Objects via a SYSTEM-PARAMETER

INTERVAL	SYSTEM PARAMETER	FREQUENCY	VALUE
INPUTS	CONSISTS OF	HAPPENS/ {TIMES} {EVERY} {[WITHIN]AFTER}	
OUTPUTS	CONSISTS OF	HAPPENS/ {TIMES} {EVERY} {[WITHIN]AFTER}	
SETS	CARDINALITY CONSISTS OF	VOLATILITY-SET * VOLATILITY-MEMBER *	
ENTITIES	CARDINALITY CONSISTS OF	VOLATILITY *	
RELATION	CONNECTIVITY CARDINALITY		
GROUPS	CONSISTS OF		
ELEMENTS			VALUE
PROCESSES		HAPPENS/ {TIMES} {EVERY} {[WITHIN]AFTER}	
EVENTS		HAPPENS/ {TIMES} {EVERY} {[WITHIN]AFTER}	
INTERVAL	CONSISTS OF	CONSISTS OF	
SYSTEM-PARAMETER			VALUE

* comment entry

Table 2.5.3
URL Statements Related to Size and Volume

2.5.5 System Size Outputs

To obtain information specifically about one or more SYSTEM-PARAMETERS, the FORMATTED PROBLEM STATEMENT may be generated. Since very few of the statements involving SYSTEM-PARAMETERS have complementary statements, much of the information presented in the FORMATTED PROBLEM STATEMENT will be in comment format.

2.5.6 System Size Completeness Checks

The following checks can be made:

1) Every INPUT should have a HAPPENS/TIMES statement.
2) Every OUTPUT should have a HAPPENS/TIMES statement.
3) Every SET should have a CARDINALITY statement.
4) Every ENTITY should have a CARDINALITY statement.
5) Every PROCESS should have a HAPPENS/TIMES statement.
6) Every EVENT should have a HAPPENS/TIMES statement.
7) Every INTERVAL should be used in some statement.
8) Every SYSTEM-PARAMETER should be used in some statement.

2.6 System-Dynamics

The description of the contents of INPUTS, OUTPUTS, ENTITIES, GROUPS and structures of PROCESSES, and the relationships among these objects produced up to this point, gives a "static" description of the system. This does not in itself state the requirements for the dynamic behavior of a system. To do this, one must describe those inputs, conditions and events which may influence what processing is performed, or the order in which it is performed.

2.6.1 System Dynamics Objects

CONDITION - a statement which can be in one of two states, TRUE or FALSE (YES or NO, etc.). The statement is given a unique name.

EVENT - an object used to describe a happening, external or internal to the system, or an occurrence which causes something else in the system to happen.

2.6.2 System-Dynamics Relationships

CAUSES/CAUSED

An EVENT or INPUT, or a CONDITION BECOMING TRUE or FALSE, CAUSES an EVENT. An EVENT is CAUSED by an EVENT, an INPUT, or a CONDITION BECOMING TRUE or FALSE.

INCEPTION-CAUSES/ON INCEPTION

INCEPTION of a PROCESS CAUSES an EVENT, or an EVENT occurs ON INCEPTION of a PROCESS.

INTERRUPTS/INTERRUPTED

A PROCESS, EVENT or INPUT, or a CONDITION BECOMING TRUE or FALSE, INTERRUPTS a PROCESS. A PROCESS is INTERRUPTED by a PROCESS, EVENT or INPUT, or by a CONDITION BECOMING TRUE or FALSE.

MAKES/MADE

An EVENT, INPUT or PROCESS MAKES a CONDITION TRUE or FALSE. A CONDITION is MADE TRUE or FALSE by an EVENT, INPUT or PROCESS.

TERMINATES/TERMINATED

A PROCESS, EVENT or INPUT, or a CONDITION BECOMING TRUE or FALSE, TERMINATES a PROCESS. A PROCESS is TERMINATED by a PROCESS, EVENT or INPUT, or by a CONDITION BECOMING TRUE or FALSE.

TERMINATION-CAUSES/ON TERMINATION

TERMINATION of a PROCESS CAUSES an EVENT, or an EVENT occurs ON TERMINATION of a PROCESS.

TRIGGERS/TRIGGERED

A PROCESS, EVENT or INPUT, or a CONDITION is BECOMING TRUE or FALSE, TRIGGERS a PROCESS. A PROCESS is TRIGGERED by a PROCESS, EVENT or INPUT, or by a CONDITION'S BECOMING TRUE or FALSE.

Conditional actions for each of the above relationships can be described by the use of the DEPENDING ON clause. Also, repetition of conditional actions can be described by the use of the FOR EACH clause.

WHILE

A CONDITION may be TRUE WHILE or FALSE WHILE some criteria hold.

2.6.3 System Dynamics Syntax and Semantics

The objects and relationships involved in describing system dynamics are shown pictorially in Figure 2.6.3 and in tabular form in Table 2.6.3.

INCEPTION or TERMINATION of a PROCESS may CAUSE any number of EVENTS. Similarly, an EVENT may occur ON INCEPTION or ON TERMINATION of any number of PROCESSES. The INCEPTION of a PROCESS is its beginning, TERMINATION is the completion of the PROCESS.

Any number of EVENTS, INPUTS, and/or CONDITIONS may CAUSE and EVENT. However, a separate statement is required for each CONDITION involved. Similarly, any number of EVENTS may be CAUSED by a given collection of EVENTS, INPUTS, and/or CONDITIONS.

Any number of EVENTS, INPUTS and/or PROCESSES may MAKE a CONDITION TRUE or FALSE. Any number of CONDITIONS may be MADE TRUE or FALSE by a given collection of EVENTS, INPUTS and/or PROCESSES. Only one of the values, TRUE and FALSE, may be used in a given MAKES of MADE statement. The term MAKES implies setting the value of a CONDITION.

Any number of PROCESSES, EVENTS, INPUTS and/or CONDITIONS may TRIGGER, INTERRUPT or TERMINATE a given PROCESS. Any number of PROCESSES may be TRIGGERED, INTERRUPTED or TERMINATED by a given collection of PROCESSES, EVENTS, INPUTS and/or CONDITIONS. To TRIGGER a PROCESS is to initiate it. A PROCESS is INTERRUPTED if it is eligible to be resumed later, while it is TERMINATED if it is ended (whether complete of not) and is not to be resumed.

A CONDITION may only have one WHILE statement, which is expressed as a comment entry. Should more than one be specified for a given CONDITION, the comment entries will be combined (the second added to the end of the first and so on).

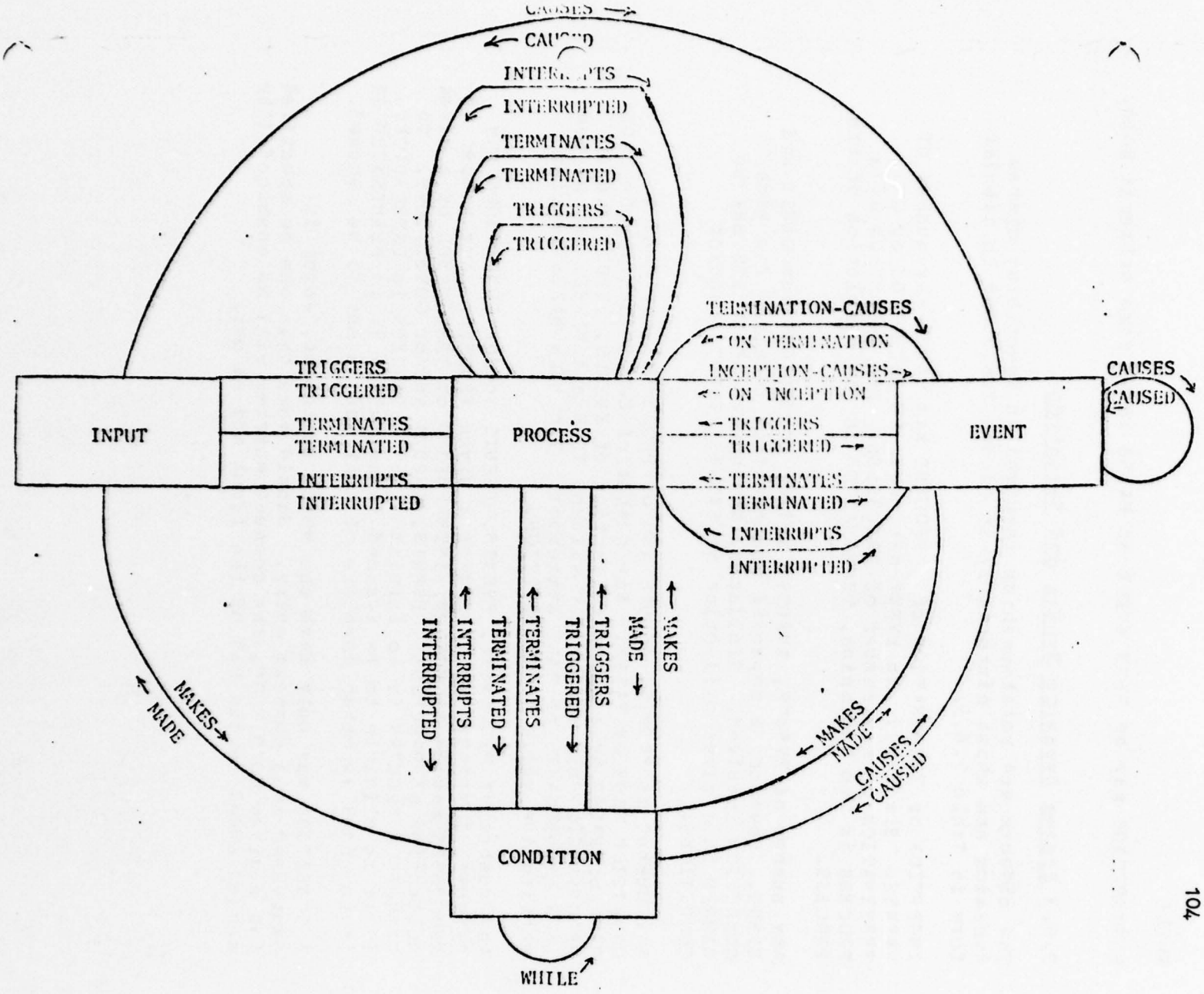


Figure 2.6.3 - System-Dynamics Objects and Relationships

	PROCESS	EVENT
PROCESS	TRIGGERS[1]	INCEPTION-CAUSES[1]
	TERMINATES[1]	TERMINATION-CAUSES[1]
	INTERRUPTS[1]	
	TRIGGERED BY[1]	TRIGGERED BY[1]
	TERMINATED BY[1]	TERMINATED BY[1]
	INTERRUPTED BY[1]	INTERRUPTED BY[1]
EVENT	TRIGGERS[1]	CAUSES[1]
	TERMINATES[1]	
	INTERRUPTS[1]	
	ON INCEPTION OF[1]	CAUSED BY[1]
	ON TERMINATION OF[1]	
CONDITION	BECOMING {TRUE} TRIGGERS {FALSE}[1]	BECOMING {TRUE} CAUSES {FALSE}[1]
	BECOMING {TRUE} TERMINATES {FALSE}[1]	
	BECOMING {TRUE} INTERRUPTS {FALSE}[1]	
	{TRUE} MADE {FALSE} BY[1]	{TRUE} MADE {FALSE} BY[1]
INPUT	TRIGGERS[1]	CAUSES[1]
	TERMINATES[1]	
	INTERRUPTS[1]	

Table 2.6.3
URL Statements for Describing System-Dyanmics

[1] Conditional (DEPENDING ON) and repetition (FOR EACH clauses are allowed.

CONDITION		INPUT
PROCESS		
MAKES --	{TRUE} {FALSE}[1]	
TRIGGERED WHEN --BECOMES	{TRUE} {FALSE}[1]	TRIGGERED BY[1] TERMINATED BY[1] INTERRUPTED BY[1]
TERMINATED WHEN --BECOMES	{TRUE} {FALSE}[1]	
INTERRUPTED WHEN --BECOMES	{TRUE} {FALSE}[1]	
EVENT		
MAKES --	{TRUE} {FALSE}[1]	
CAUSED WHEN -- BECOMES	{TRUE} {FALSE}[1]	CAUSED BY[1]
CONDITION	WHILE[2]	MADE {TRUE} {FALSE} BY[1]
INPUT		
MAKES --	{TRUE} {FALSE}[1]	

[1] Conditional (DEPENDING ON) and repetition (FOR EACH clauses are allowed.

[2] comment entry only

Table 2.6.3 (Continued)

2.6.4 System Dynamics Common Equivalents and Usage

As is the case with system size, description of system dynamics aspects are often not stated explicitly but are incorporated into the descriptions of other objects. In some cases, this type of information is presented by decision tables or by decision blocks in flow charting methods.

Since decision tables present a plan of "action" based on conditions and events, they may be given in the PROCEDURE statement for the appropriate PROCESS, if desired.

A list of EVENTS TRIGGERING a PROCESS implies that each one of the EVENTS TRIGGERS the PROCESS. Since an EVENT occurs at an instant in time, the user should not need to say that a combination of EVENTS TRIGGERS a PROCESS, since this would require that all the EVENTS occur simultaneously.

Even though there is no way to state explicitly that a combination of CONDITIONS TRIGGERS a PROCESS, this may easily be handled by defining a new CONDITION to represent the combination. For example, if PROCESS P1 is TRIGGERED when CONDITION C1 is TRUE and CONDITION C2 is FALSE, the user may write:

```
CONDITION C3;
   TRUE WHILE;
      C1 AND NOT C2;

PROCESS P1;
   TRIGGERED WHEN C3 BECOMES TRUE;
```

Any EVENT or CONDITION that affects the system's operation, should be defined.

2.6.5 System Dynamics Outputs

The FORMATTED PROBLEM STATEMENT may be generated to obtain information about one or more CONDITIONS or EVENTS.

The PROCESS CHAIN report will show structures of EVENTS and PROCESSES connected by TRIGGERS and TRIGGERED BY statements.

2.6.6 System Dynamics Completeness Checks

1) Every EVENT should be associated with at least one CONDITION or PROCESS.
2) Every CONDITION should be associated with at least one EVENT or PROCESS.
3) Every CONDITION should have a TRUE WHILE or a FALSE WHILE statement.

2.7 System Architecture

The system architecture description deals with the physical aspects of an information processing system.

2.7.1 System Architecture Objects

PROCESSOR - an object that can "perform" a PROCESS.

RESOURCE - something that the physical elements in the target system consume in order to carry out information processing functions.

UNIT - an object used to measure RESOURCES.

RESOURCE-USAGE-PARAMETER - used to define a measure of the RESOURCE usage for a PROCESS.

2.7.2 System Architecture Relationships

CONSUMES/CONSUMED BY - A RESOURCE may be CONSUMED BY a PROCESSOR, and a PROCESSOR May CONSUME an amount of RESOURCE PER RESOURCE-USAGE-PARAMETER.

PERFORMS/PERFORMED BY - A PROCESSOR may PERFORM a PROCESS, and a PROCESS may be PERFORMED BY a PROCESSOR.

MEASURES/MEASURED IN - A UNIT may MEASURE a RESOURCE, and a RESOURCE may be MEASURED IN a UNIT.

RESOURCE-USAGE/RESOURCE-USAGE-PARAMETER-VALUE - A PROCESS may have a RESOURCE-USAGE-PARAMETER-VALUE associated with a RESOURCE-USAGE-PARAMETER.

2.7.3 System Architecture Syntax and Semantics

The objects and relationships involved in describing system architecture are shown in Table 2.7.3.

A PROCESS may have an arbitrary number of RESOURCE-USAGE-PARAMETER and RESOURCE-USAGE-PARAMETER-VALUE pairs. (But there can only be at most one such pair for a particular RESOURCE-USAGE-PARAMETER.) This pair is used to describe the expected resource consumption by the execution of the PROCESS in a PROCESSOR independent manner. The CONSUMES statement in the PROCESSOR section specifies the name and amount of RESOURCES that are consumed per RESOURCE-USAGE-PARAMETER of the PROCESS it performs. This measure is translated to a resource consumption

value by multiplying the RESOURCE-USAGE-PARAMETER-VALUE with the resource-consumption-value for the RESOURCE-USAGE-PARAMETER in the CONSUMES statement of the PROCESSOR. For example, suppose that there is a PROCESS "P1," and a PROCESSOR "PR1," and that the RESOURCE in question is "CPU-TIME" (measured in UNIT of "MICRO-SECONDS"), as in Figure 2.7.3.

PROCESSOR	RESOURCE	UNIT	RESOURCE-USAGE-PARAMETER	PROCESS
PROCESSOR	SUBPARTS PART OF	CONSUMES	CONSUMES PER	PERFORMS
RESOURCE	CONSUMED BY	MEASURED IN		
UNIT		MEASURES		
RESOURCE USAGE PARAMETER				RESOURCE-USAGE-PARAMETER-VALUE FOR
PROCESS	PERFORMED		RESOURCE-USAGE	

Table 2.7.3
System Architecture Relationships

```
PROCESS P1;
   RESOURCE-USAGE:  100 FOR NO-OF-STATEMENT;

PROCESS P2;
   RESOURCE-USAGE:  200 FOR NO-OF-STATEMENT

PROCESSOR PR1;
   PERFORMS P1;
   CONSUMES CPU-TIME AT RATE OF 20 PER NO-OF-
     STATEMENTS;
```

Figure 2.7.3
Example of URL statements for
PROCESSOR and its RESOURCE-usage.

Here "NO-OF-STATEMENT" is a RESOURCE-USAGE-PARAMETER. The PROCESS called P1 has a value of 100 for this parameter. One possible interpretation of this statement is that the relative

difficulty or complexity of the PROCESS is such that it would take 100 "statements" on a hypothetical processor. Other PROCESSes may be given values for the same RESOURCE-USAGE-PARAMETER. For example, PROCESS P2, which is considered twice as difficult or complex, is given the value 200 for this RESOURCE-USAGE-PARAMETER. Note that the RESOURCE-USAGE-PARAMETER and its value are meant to be PROCESSOR independent. They are used to record estimation of RESOURCE-USAGE independent of what PROCESSOR performs the particular PROCESS.

In the PROCESSOR section, the CONSUMES statement is used to record the resource-consumption-value for a RESOURCE-USAGE-PARAMETER. In the example of Figure 2.7.3, 20 is the resource-consumption-value of the PROCESSOR "PR1" for the RESOURCE-USAGE-PARAMETER "NO-OF-STATEMENT." "MICRO-SECONDS" is the name of the UNIT that is used to measure the RESOURCE called "CPU-TIME."

This statement may be interpreted as saying that the PROCESSOR "PR1" will consume 20 microseconds of CPU time per "number of statements" (given in the PROCESS description) whenever it performs a PROCESS. In this example, 2,000 (100 x 20) microseconds of CPU time is consumed by PROCESSOR "PR1" whenever it performs PROCESS "P1," and 4,000 (200 x 20) microseconds for "P2."

It is possible to associate more than one RESOURCE-USAGE-PARAMETER (and its value) for a PROCESS. It may be used to allow for the possibility of employing two completely different types of processors (like a computer and a person) to perform the PROCESS. In this way, the decision as to what PROCESSOR to use for a particular PROCESS may be delayed as necessary and changing the PROCESSOR for a PROCESS once it is decided is easier. Having more than one pair of RESOURCE-USAGE-PARAMETERs and its value may also be used to describe resource consumption independently for more than one resource. Only the resource consumption value, which has the same RESOURCE-USAGE-PARAMETER in both PROCESS and PROCESSOR sections, is taken as contributing to the actual resource consumption. If there are multiple instances of such PARAMETERs, the net consumption for a resource is the sum of all the consumption values.

The PERFORMS/PERFORMED BY statement is to record the relationship between a PROCESS and the PROCESSOR that performs (i.e., carries out, does, etc.) the PROCESS. A PROCESSOR can perform more than one PROCESSes, but a PROCESS can be performed by only one PROCESSOR.

The MEASURES/MEASURED IN statement is to define relationships between a UNIT and a RESOURCE. A UNIT may measure more than one RESOURCE, but a RESOURCE can be measured only in one UNIT. The UNIT name that appears after the resource-consumption-value in the CONSUMES statement of the PROCESSOR section is optional, but if it is given it must be the correct UNIT name for that

RESOURCE.

2.7.4 System Architecture Completeness Checks

The completeness checks that can be made for SAF objects are:

1) Every PROCESS should be PERFORMED BY a PROCESSOR and every PROCESSOR should PERFORM at least one PROCESS. At each subdivision of PROCESS and PROCESSOR SUBPARTS/PART OF structure, the PERFORMS/PERFORMED BY relationships of the subparts should be consistent with the relationships of the parent objects.

2) If a PROCESSOR PERFORMS a PROCESS, at least one common RESOURCE-USAGE-PARAMETER must be defined for the PROCESSOR (via CONSUMES statement), and for the PROCESS (via RESOURCE-USAGE statement).

3) If a SYSTEM-PARAMETER is used for RESOURCE-USAGE-PARAMETER-VALUE or in the CONSUMES statement of the PROCESSOR section, it must have a single numerical value.

4) Every UNIT should MEASURE at least one RESOURCE, and every RESOURCE should be measured in a UNIT, and CONSUMED BY at least one PROCESSOR.

2.8 Properties

The facilities described in this section are available to aid all aspects of documentation, communication and analysis. These facilities also provide open-ended classification systems since these "qualifiers" may be added at any time and used for retrieval of parts of the problem statement. They can be used to describe any of the objects whether in the organization, the target system or in the project. They may be used in cases where the analyst wishes to include some information in the documentation where no formal syntax is available.

2.8.1 Properties Objects

SYNONYM - is used to define an alternative name (alias) for a given named object in the URL description of the system.

KEYWORD - an object associated to one or more names for the purpose of selection and analysis.

MEMO - an object which represents text relevant to one or more other objects.

ATTRIBUTE and ATTRIBUTE-VALUE - objects used to describe characteristics of objects not otherwise allowed in the language.

SOURCE - an object which is to be referenced for more information about an object. Examples of SOURCES are interview-reports, company procedure manuals, documents, etc.

SECURITY - an object which identifies what points of the problem statement may be reviewed by what individuals.

TRACE-KEY - an object which is used to correlate objects which exist in different data bases.

2.8.2 Properties Relationships

DESCRIPTION

Any object defined in the problem statement may have a DESCRIPTION, which consists of one or more lines of narrative text. A DESCRIPTION is not a URL object and does not have a URL name.

SYNONYM

Any type of object may have SYNONYMS and a SYNONYM may be DESIGNATED for a given object.

ASSERT

Any object which has a relationship with another object may have an ASSERT statement. An ASSERT statement asserts that one object must have a particular ATTRIBUTE and ATTRIBUTE-VALUE when related to another object.

ATTRIBUTES

Any object may have ATTRIBUTES with corresponding ATTRIBUTE-VALUES.

KEYWORDS/APPLIES

Any object may have KEYWORDS associated with it and a KEYWORD may APPLY to any type of object.

SEE-MEMO/APPLIES

Any object may have a SEE-MEMO and a MEMO may APPLY to any object.

SOURCE/APPLIES

Any object may have a SOURCE, and a SOURCE may APPLY to any object.

SECURITY/APPLIES

Any object may have a SECURITY, and a SECURITY may APPLY to any object.

TRACE-KEY/APPLIES

Any object may have a TRACE-KEY, and a TRACE-KEY may APPLY to any object.

2.8.3 Properties Syntax and Semantics

The objects and relationships involved in describing properties are shown pictorially in Figure 2.8.3 and in tabular form in Table 2.8.3.

A given object may have only one DESCRIPTION. If more than one DESCRIPTION is specified, they will be combined (concatenated to the end of the previously specified DESCRIPTION). When entering a DESCRIPTION into the data base it is important to note that the text must start on the line following the word DESCRIPTION.

A given object may have any number of SYNONYMS, but a given SYNONYM may belong to only one object. SYNONYMS may be used anywhere in a Problem Statement that the basic name may be used. A SYNONYM must be a URL name.

A given object may have any number of KEYWORDS associated with it. KEYWORDS, however, may not have KEYWORDS. A KEYWORD may APPLY to any number of object names.

A given object may have any number of ATTRIBUTES, but for a given ATTRIBUTE, may only have one ATTRIBUTE-VALUE. ATTRIBUTES may not have ATTRIBUTES. An ATTRIBUTE can have any number of values.

A given object may have any number of ASSERT statements which relate that object to other objects having particular ATTRIBUTES and ATTRIBUTE-VALUES.

A given object may have any number of SEE-MEMO statements. A MEMO, however, may not have any SEE-MEMO statements. A MEMO may APPLY to any number of named objects.

An object may have any number of SOURCES and any SOURCE may APPLY to any number of objects.

An object may have any number of SECURITIES and any SECURITY may APPLY to any number of objects.

An object may have any number of TRACE-KEYS and any TRACE-KEY may APPLY to any number of objects.

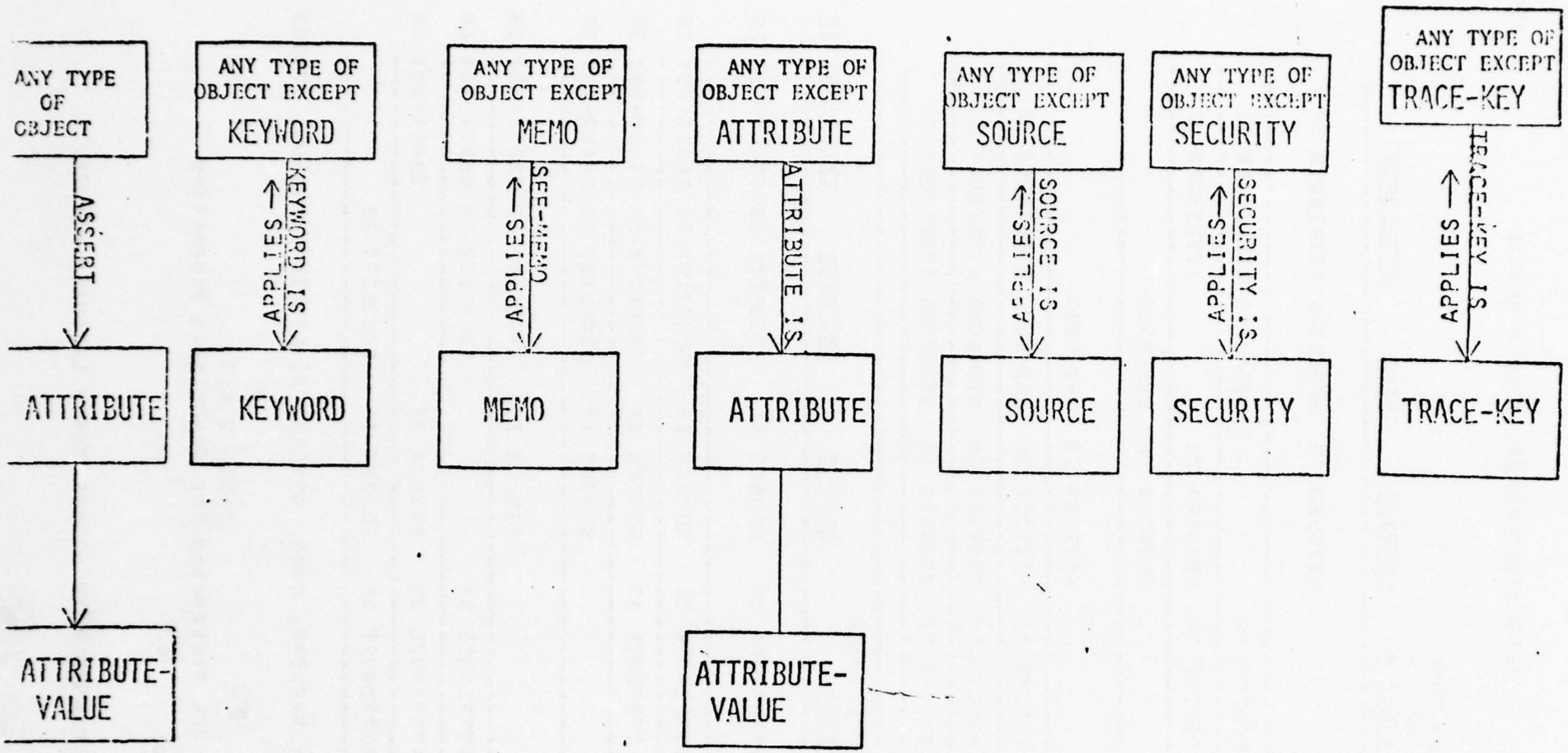


FIGURE 2.8.3 URL STATEMENTS DESCRIBING PROPERTIES

Any Type of Object *	KEYWORD	MEMO	ATTRIBUTE
Any Type of Object *	KEYWORD IS	SEE-MEMO	ATTRIBUTE IS
KEYWORD	APPLIES TO	SEE-MEMO	ATTRIBUTE IS
MEMO	APPLIES TO	KEYWORD IS	ATTRIBUTE IS
ATTRIBUTE	KEYWORD IS	SEE-MEMO	
ATTRIBUTE VALUE	KEYWORD IS	SEE-MEMO	
SOURCE	APPLIES TO	KEYWORD IS	SEE-MEMO ATTRIBUTE IS
SECURITY	APPLIES TO	KEYWORD IS	SEE-MEMO ATTRIBUTE IS
TRACE-KEY	APPLIES TO	KEYWORD IS	SEE-MEMO ATTRIBUTE IS

Any Type of Object	ATTRIBUTE-VALUE	SOURCE	SECURITY	TRACE-KEY IS
Any Type of Object	ATTRIBUTE IS	SOURCE IS	SECURITY IS	TRACE-KEY IS
KEYWORD	ATTRIBUTE IS	SOURCE IS	SECURITY IS	TRACE-KEY IS
MEMO	ATTRIBUTE IS	SOURCE IS	SECURITY IS	TRACE-KEY IS
ATTRIBUTE		SOURCE IS	SECURITY IS	TRACE-KEY IS
ATTRIBUTE VALUE		SOURCE IS	SECURITY IS	TRACE-KEY IS
SOURCE	ATTRIBUTE IS		SECURITY IS	TRACE-KEY IS
SECURITY	ATTRIBUTE IS	SOURCE IS		TRACE-KEY IS
TRACE-KEY	ATTRIBUTE IS	SOURCE IS	SECURITY IS	

* other than KEYWORD, MEMO, ATTRIBUTE, ATTRIBUTE-VALUE, SOURCE or SECURITY

Table 2.8.3
URL Statements for Describing Properties

2.8.4 Properties Common Equivalents and Usage

The DESCRIPTION associated with a given object is analogous to any text description presented in most documentation methods. It may contain any tables, charts or figures which can be displayed by the output device.

A URL SYNONYM has the same meaning as commonly used. Its two major uses in URL are:

1) To reduce the number of characters used in specifying the problem statement. This can be accomplished by assigning a very short SYNONYM to each user defined name as it is defined.

2) To allow different problem definers to reference the same object by different names.

KEYWORDS may be used to logically group several objects for retrieval and analysis purposes. For example, to generate URA reports for only those PROCESSES which were to run in batch mode, each of the PROCESSES could have the following KEYWORD statement:

```
KEYWORD:  BATCH ;
```

Using the KEY= facility in the NAME-GEN command, all the PROCESSES with a KEYWORD 'BATCH' could be retrieved. Any desired outputs could be produced by URA at this point.

ATTRIBUTES may also be thought of as qualifiers. For example, to present mode and length information about an ELEMENT, the following ATTRIBUTES statement might be used:

```
ATTRIBUTES:  MODE NUMERIC,
             LENGTH 8  ;
```

The ATTRIBUTE statement can be used to fill any number of requirements for specifying characteristics of objects. For PROCESSES, processing mode, duration might be given; for INPUTS and OUTPUTS, format or size might be given; etc.

The ASSERT statement may be used to present more information about an existing relationship. For example, if:

```
PROCESS:  get-names DERIVES name USING number;
```

an appropriate ASSERT relationship would be:

```
ASSERT name type char, number type integer;
```

URL provides the facility in KEYWORD and ATTRIBUTE statements for the classificatior of objects by a criteria which can be defined and expanded as the project progresses. The additional

information can be added at any time during the project without disturbing the data gathered up to that point.

SECURITY and SOURCE refer to the definition of the objects, not to the security of data or source of data in the target system.

The TRACE-KEY statement is used to correlate objects contained in different data bases. The security level in a logical system design data base and a security level number in a physical system design data base may both have the statement:

TRACE-KEY: security-level-key;

2.8.5 Properties Outputs

The DICTIONARY report presents SYNONYMS, the DESCRPITION and KEYWORDS for each name given as input.

The NAME-GEN command can retrieve all names with a particular KEYWORD value by using the KEY parameter. Reports may then be generated for the selected names by utilizing the default facilities of URA.

The ATTRIBUTE report presents information about ATTRIBUTES in the problem statement by presenting those objects the particular ATTRIBUTES are associated with and corresponding ATTRIBUTE-VALUES.

2.8.6 Properties Completeness Checks

None of the properties are "necessary" for a complete description as it is up to the organization to impose any requirements to what type of properties are to be incorporated in the documenation.

However, every property object defined should be used at least once.

1) Every KEYWORD should APPLY to at least one object.
2) Every ATTRIBUTE should APPLY to at least one object.
3) Every MEMO should APPLY to at least one object.
4) Every SOURCE should be the source for at least one object.
5) Every SECURITY should be referenced in at least one object.
6) Every TRACE-KEY should be referenced by at least one object.

2.9 Project Management

All object and statement facilities in URL/URA, which are intended to improve organization and management within the project and present information about the project describing the system, is referred to as Project Management.

2.9.1 Project Management Objects

PROBLEM-DEFINER - an object responsible for the URL description of one or more of the objects being described. Usually, the URL names will be the name of a person in the form normally used in the organization.

MAILBOX - an object which identifies an address by which information may be sent to a particular PROBLEM-DEFINER. In time sharing systems, which provide such a service, the MAILBOX would be the PROBLEM-DEFINER's ID.

2.9.2 Project Management Relationships

RESPONSIBLE-PROBLEM-DEFINER/RESPONSIBLE FOR

A PROBLEM-DEFINER may be RESPONSIBLE for the description of any other object, and any object may have a RESPONSIBLE-PROBLEM-DEFINER.

MAILBOX/APPLIES

A PROBLEM-DEFINER may have a MAILBOX and a MAILBOX may APPLY to a PROBLEM-DEFINER.

2.9.3 Project Management Syntax and Semantics

The objects and relationships involved in describing he project management aspect of a system are shown pictorially in Figure 2.9.3 and in tabular form in Table 2.9.3.

The RESPONSIBLE-PROBLEM-DEFINER statement implies that the given PROBLEM-DEFINER accepts responsibility for the URL description of the designated object: it is assumed that any questions concerning this description can be handled by the PROBLEM-DEFINER. A given object may have only one RESPONSBILE-PROBLEM-DEFINER, but a PROBLEM-DEFINER may be RESPONSIBLE for many object descriptions.

A PROBLEM-DEFINER may have only one MAILBOX, but a MAILBOX may

APPLY to any number of PROBLEM-DEFINERS.

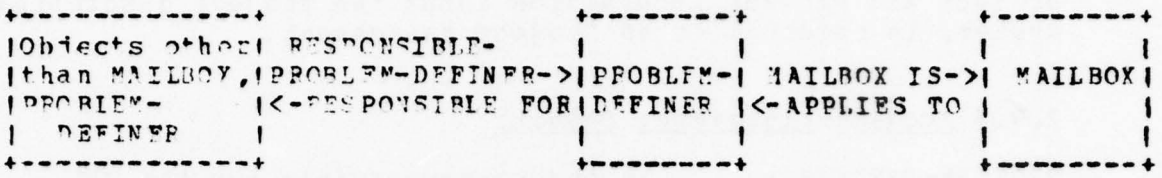


Figure 2.9.3 URL Statements for Describing Project Management

Other Objects Except Problem Definer	Problem Definer	Mailbox
Other Objects	RESPONSIBLE-PROBLEM-DEFINER	
Problem Definer	RESPONSIBLE FOR	MAILBOX IS
Mailbox	APPLIES TO	

Table 2.9.3
URL Statements for Describing Project Management

2.9.4 Project Management Common Equivalents and Usage

The meaning of these terms are the same as those in common use. These statements are intended to help the project management. The implementation (i.e., their use in a particular project) depends on the particular situation and the standards in use in the organization.

2.9.5 Project Management Outputs

Information relevant to project management can be presented in a FORMATTED PROBLEM STATEMENT for appropriate PROBLEM-DEFINERS and MAILBOXES.

The DATA BASE SUMMARY report gives the number of each objects of each type that have been defined, and how many have SYNONYMS and DESCRIPTIONS. This report can be used by the project leader to review the degree of progress in the project.

2.9.6 Project Management Completeness Checks

1) Every PROBLEM-DEFINER should be RESPONSIBLE for at least one object.
2) Every object should have one and only one RESPONSIBLE-PROBLEM-DEFINER.
3) Every MAILBOX should APPLY to at least one PROBLEM-DEFINER.
4) Every PROBLEM-DEFINER should have a MAILBOX.

3. URL SYNTAX AND SEMANTICS BY TYPE OF OBJECTS

The full and detailed syntax of URL is contained in Part II of this document. There, Section 3 contains a summary of the statements in each section with the sections in alphabetical order. Section 4 contains the description of each statement. Within a section, statements appear in alphabetical order by statement name.

In this section the Sections and Statements are presented in a different order. The paragraphs following each statement describe the statement and give the syntax for each statement and an example of their usage.

As in Section 2, the explanations of URL statements include three levels of precision:

"must" -	denotes that this is checked by URA and not entered into the data base unless correct.
"should" -	denotes that this is not checked by URA before stored in the data base but is necessary for a complete description of the target system. Some of these "completeness" checks are made when producing URA reports and warning messages are produced. Others can be made by the analyst using URA reports.
"implies" - and "may"	denotes the semantic meaning of the statement. This is not checked by URA nor necessary for a complete description. Interpretation is to be decided by the Problem Definer and organization.

The URL reserved word in parentheses after the syntax notation for a statement, specifies an acceptable abbreviation for the long form of the statement's reserved word(s).

The word "section" is used in URL to denote a number of statements and in this paper to denote a number of paragraphs. To avid confusion, the fist letter will be capitalized when referring to a URL Section.

3.1 Order of Presentation

3.1.1 Order of the Section

The rest of Section 3 specifies the complete syntax of the statements for each URI Section. The URL Sections are presented in the order shown in Table 3.1.1.

3.1.2 Order of Statements Within a Section

The facilities of URL to state an information processing problem have been described in section 2 in order by a sequence of different aspects. The particular sequence chosen is a natural one in which to learn the language. It is also a natural one when the problem is being defined in top-down fashion. In this section, within each URL Section description, the corresponding URL statements are ordered according to the aspect of the system description to which the Statements apply. The aspects of the system description are given in the following order:

System Flow
System Structure
Data Structure
Data Derivation
System Size
System Dynamics
System Architecture
System Properties
Project Management

Since System Property and Project Management statements can appear in almost every section, they are given only once in 3.2.

Regardless of the order in which statements are entered into the URA data base, they appear in the FORMATTED PROBLEM STATEMENT in a standard order. The order is essentially that followed in section 2 and summarized in Table 3.1.1. (The order in which the sections (i.e., the types of objects) appear in the report is the one in which the types of objects were listed in the file used as the input to the NAME-GEN command and to produce the FORMATTED PROBLEM STATEMENT.)

INTERFACE or REAL-WORLD-ENTITY	3.3
INPUT	3.4
OUTPUT	3.5
ENTITY	3.6
SET	3.7
RELATION	3.8
GROUPS and ELEMENTS	3.9
PROCESS	3.10
INTERVAL	3.11
CONDITION	3.12
EVENT	3.13
PROCESSOR	3.14
RESOURCE	3.15
RESOURCE-USAGE-PARAMETER	3.16
UNIT	3.17
PROBLEM-DEFINER	3.18
MEMO	3.19
DEFINE	3.20
ATTRIBUTE	
ATTRIBUTE-VALUE	
CLASSIFICATION	
KEYWORD	
MAILBOX	
SECURITY	
SOURCE	
SUBSETTING-CRITERION	
SYSTEM-PARAMETER	
TRACE-KEY	
DESIGNATE	3.21
SYNONYM	

Table 3.1.1 Order of URL Section

3.2 Statements Permitted in Almost Every URL Section

The URL statements that may be allowed in a given URL Section are dependent on the types of objects defined by the section header. Where it is illogical to say that an ELEMENT USES a PROCESS, to state that a PROCESS USES an ELEMENT would be allowed.

There are, however, the URL statements related to System Properties and Project Management that can be used within almost any Section. These statements are described in this subsection.

3.2.1 SYNONYM Statement

SYNONYMS are alternative names, or abbreviations, that may be used to reference a particular object name. SYNONYMS must be unique within the problem statement, though an object can have any number of SYNONYMS.

Syntax:

SYNONYMS (SYN) _____:
(list of synonym names)

Example:

For a long name like "departments-and-employees," it may be easier to reference it by specifying short synonyms:

SYNONYMS: dept-emp, de;

3.2.2 DESCRIPTION Statement

The DESCRIPTION statement allows the problem definer to specify information about an object in a narrative format. There are no restrictions on what is allowed in the narrative description except that a semi-colon cannot be used inside since it is used to denote the end of the statement. Any number of DESCRIPTION statements may be given for an object, but all are combined into one DESCRIPTION. Any subsequent DESCRIPTIONS are concatenated to the current DESCRIPTION.

Syntax:

DESCRIPTION (DESC):

_____
_____
_____
_____ ;
(narrative description)

Example:

To describe the highest level PROCESS in the system being described, the following DESCRIPTION statement may be applicable:

DESCRIPTION;

This is the highest level process. It accepts all input to the system and produces all outputs. ;

3.2.3 KEYWORD Statement

The KEYWORD statement can be used to logically relate object names together for retrieval and subsequent analysis purposes. An object may have any number of KEYWORDS.

Syntax:

KEYWORD (KEY) _____:
(list of keyword names)

Example:

The following statement may be used to identify particular PROCESSES as lowest-level processes:

KEYWORD: TERMINAL;

All PROCESSES with the KEYWORD "TERMINAL" can be subsequently retrieved together and analyzed in available URA reports.

3.2.4 ATTRIBUTES Statement

ATTRIBUTES are used to state specific characteristics of given objects. The ATTRIBUTE name designates the name of the characteristic and the ATTRIBUTE-VALUE, the value or magnitude of this characteristic. The ATTRIBUTE-VALUE may be either a URL name or an integer.

An object may have any number of ATTRIBUTES. A given ATTRIBUTE can refer to any number of objects not necessarily of the same type.

Syntax:

ATTRIBUTES (ATTR) _____ _____,
attribute name attribute-value name

_____ _____,
attribute name attribute-value name

_____ _____;
attribute name attribute-value name

Example:

To specify that a particular data element is numeric field of length six, the following statement may be used:

ATTRIBUTES: TYPE NUMERIC,
LENGTH SIX ;

3.2.5 ASSERT Statement

The ASSERT statement allows the Problem Definer to assert that one object must have a particular ATTRIBUTE and ATTRIBUTE-VALUE when related to another object. An object may have a number of ASSERT statements.

Syntax:

ASSERT (ASRT) _____;
(list of names followed by attributes and attribute-values)

Example:

If PROCESS get-name DERIVES name USING number, an appropriate ASSERT statement would be:

```
ASSERT:  name type char,
         number type integer;
```

3.2.6 RESPONSIBLE-PROBLEM-DEFINER Statement

The RESPONSIBLE-PROBLEM-DEFINER statement specifies that one problem definer person is responsible for initial preparation and/or maintenance of an object description. Only one problem definer may be delegated responsibility for a given Section, but may be responsible for more than one Section.

Syntax:

RESPONSIBILE-PROBLEM-DEFINER (RPD) _____;
(name of responsible-problem-definer)

Example:

To specify that Michel Bastarache is responsible for the URL description for a particular object, state:

```
RESPONSIBLE-PROBLEM-DEFINER MICHEL-BASTARACHE;
```

in the URL Section for that object.

3.2.7 SEE-MEMO

The SEE-MEMO statement allows a description common to several objects (and available in a MEMO's DESCRIPTION) to be referenced. This statement may occur any number of times for a given object.

Syntax:

SEE-MEMO (SM) _____;
(list of memo names)

Example:

To refer to a particular MEMO on programming conventions relevant to the description of low level PROCESSES, the following may be given:

SEE-MEMO: PROGRAMMING-CONVENTIONS;

3.2.8 SOURCE Statement

The SOURCE statement identifies information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc. Any number of SOURCES may be related to an object.

Syntax:

SOURCE (SRC) _____;
(list of source names)

Example:

To make reference to a paper written by Constantine:

SOURCE: CONSTANTINE;

The URL description of the SOURCE name, CONSTANTINE, would probably specify relevant information such as name of paper, date published, etc.

3.2.9 SECURITY Statement

The SECURITY statement specifies the level of security associated with a given object's URL description. Any number of SECURITIES may be related to an object.

Syntax:

SECURITY (SEC) _____;
(list of security names)

Example:

To specify that the URL description for a given object may only be viewed by company personnel, the following statement may be used:

SECURITY: COMPANY;

3.2.10 TRACE-KEY Statement

A TRACE-KEY is used to correlate objects which exist in different data bases. An object may have several TRACE-KEYS.

Syntax:

TRACE-KEY (TKEY) _____:
(list of trace-key names)

Example:

The security level in a logical system design data base and a security level number in a physical system design data base may both have the statement:

TRACE-KEY: security-level-key;

3.3 INTERFACE Section

REAL-WORLD-ENTITIES or INTERFACES are named objects, outside the target system, that interact with the system being described. If the system being described was a payroll system, one possible INTERFACE would be the employees paid by the system. They could be, in one sense, the customers of the system.

INTERFACE (INTF) _____;
(list of interface names)

3.3.1 System-Flow Statements for INTERFACES

The RECEIVES statement is used to specify that the INTERFACE accepts information (OUTPUTS) from the target system.

RECEIVES (RCVS) _____;
(list of output names)

To specify the manner in which INTERFACE receives OUTPUTS more precisely, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the RECEIVES statement.

RECEIVES _____
(list of output names)

DEPENDING ON _____ _____
(list of elements, condition-names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

The GENERATES statement is used to specify that the INTERFACE produces information (INPUT) which is used by the system.

GENERATES (GENS) _____;
(list of input names)

To specify the manner in which INTERFACE generates INPUTS more precisely, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the GENERATES statement.

GENERATES _____
(list of input names)

DEPENDING ON _____ __
(list of element, condition-names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

The RESPONSIBLE statement specifies that an INTERFACE has the responsibility of maintaining information (SETS) within the target system.

RESPONSIBLE (RESP) _____;
(list of set names)

To insure completeness of the problem statement, the problem definer should check that every INTERFACE either GENERATES some INPUT, RECEIVES some OUTPUT or is RESPONSIBLE for some SET.

An INTERFACE, therefore, can interact with the system only through RECEIVING OUTPUTS, GENERATING INPUTS or being RESPONSIBLE FOR SETS. In particular, it is not possible to describe any processing performed in the INTERFACE. If, in the system description, it is necessary to describe processing in the INTERFACE, then it should be designated as a PROCESS instead of an INTERFACE. See section 4.1 on system boundaries.

3.3.2 System-Structure Statements for INTERFACES

An INTERFACE may be part of one, and only one, larger INTERFACE, and it may have any number of subparts that are also INTERFACES.

PART _____;
(interface name)

SUBPARTS (SUBP) _____;
(list of interface names)

These statements permit organization structures to be specified. This can be used to obtain, from URA, descriptions of the system as seen from a particular part of the organization.

3.3.3 Data-Derivation Statements for INTERFACES

In the target system, an INTERFACE may have the right to access information of certain classifications and categories.

SECURITY-ACCESS-RIGHT (SAR) _____;
(list of classification names optionally followed by classification levels)

3.3.4 Project-Management Statements for INTERFACES

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.3.5 System-Properties Statements for INTERFACES

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.4 INPUT Section

INPUTS are information that is produced (GENERATED) by INTERFACES and that is brought into (RECEIVED BY) the target system.

INPUT (INP) _____;
(list of input names)

The name of the INPUT can be considered as the name attached to either the collection of data values or the physical medium on which the data values are recorded, i.e., the carrier of the data values, or to both.

3.4.1 System-Flow Statements for INPUTS

The names of the INTERFACES providing the INPUT are given in the GENERATED statement.

GENERATED BY (GENB) _____;
(list of interface names)

To specify the manner in which an INPUT is generated more precisely, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the GENERATED BY statement:

GENERATED BY _____;
(list of interface names)

DEPENDING ON _____;
(list of element, condition-names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

The object in the system which accepts the INPUT is given in the RECEIVED BY statement:

RECEIVED BY (RCVD) _____;
(list of process names)

To specify the manner in which an INPUT is generated more precisely. The DEPENDING ON and FOR EACH clauses may be used in conjunction with the RECEIVED BY statement.

RECEIVED BY _____
(list of process names)

DEPENDING ON _____
(list of element condition-names)

FOR EACH _____;
(list of entity/input/output/group/element/set-names)

These statements refer only to the logical collection of data elements value, and provide a way of stating where the INPUT comes from and what PROCESS must accomplish whatever is necessary to "accept" it. All operations on the data element values must be specified separately in the definition of the PROCESS.

Every INPUT should be GENERATED by at least one INTERFACE and RECEIVED by at least one PROCESS.

3.4.2 System-Structure Statements for INPUTS

An INPUT may be part of one, and only one, larger INPUT, and it may have any number of subparts that are also INPUTS.

PART _____;
(name of input)

SUBPARTS (SUBP) _____;
(list of input names)

These statements allow definitional structures (grouping INPUTS together to call them by a single name) and high level data structures to be specified. The lowest level of INPUTS normally will be used for physical documents, messages, cards, etc., that flow into the system.

To describe a collection of INPUT occurrences (SET of INPUTS), the CONTAINED statement may be used to relate INPUTS to SETS.

CONTAINED (CNTD) _____;
(list of set names)

This SET can then be used in further statement of requirements. This might be used, for example, to describe a batch of inputs such as time cards which are to be treated as a unit for processing.

An INPUT can be contained in any number of SETS.

3.4.3 Data-Structure Statements for INPUTS

The data (GROUPS and ELEMENTS) whose values appear on an INPUT are defined via the CONSISTS statement. Each data name used in the statement can be optionally preceded by a SYSTEM-PARAMETER to define the number of occurrences of the data value that may appear on the INPUT. The CONSISTS statement only specifies the data on the INPUT and implies nothing about format.

CONSISTS (CSTS) _____;
(list of group and element
each name optionally
presented by a system-parameter.)

A complete problem statement should have all INPUTS (which do not have SUBPARTS statements) broken down into GROUPS and ELEMENTS.

3.4.4 Data-Derivation Statements for INPUTS

The USED statements specifies those PROCESSES which use the information available in the INPUT.

USED _____;
(list of process names)

This implies that at least one piece of data (GROUP or ELEMENT) on the INPUT is being USED. To specify the manner in which the INPUT is used more precisely, the DERIVE or UPDATE clause may be used in conjunction with the USED statement.

USED BY _____
(list of process names)

TO DERIVE (DRV) _____;
(list of element, group, entity,
set and output names)

USED BY _____
(list of process names)

TO UPDATE (UPD) _____;
(list of element, group,
entity and set names)

An INPUT may be USED by any number of PROCESSES. Every INPUT should be used by at least one PROCESS.

The CLASSIFICATION of an INPUT may be specified with the CLASSIFICATION statment:

CLASSIFICATION _____;
(list of classfication names,
each optionally followed by
a level number)

Any PROCESSES or PROCESSORS that use the INPUT must have SECURITY-ACCESS-RIGHTS that match the classification of the INPUT.

3.4.5 System-Dynamics Statements for INPUTS

More than one individual instance of an INPUT may occur over some period of time. The number of instances of the INPUT that occur over time is stated through the HAPPENS statement:

HAPPENS (HAP) _____
(system-parameter)

TIMES-PER (TIMP) _____;
(interval name)

EVERY _____ _____ ;
(system-parameter) (interval name)

[WITHIN]_____ _____AFTER_____;
(system-parameter) (interval name) (event)

Every INPUT should have a HAPPENS statement.

The arrival of an INPUT may affect the processing currently being performed, or it may initiate new processing. This is described via the TRIGGERS, TERMINATES and INTERRUPTS statements:

TRIGGERS (TRGS) _____;
(list of process names)

TERMINATES (TRMS) _____;
(list of process names)

INTERRUPTS (INTS) _____;
(list of process names)

The DEPENDING ON and FOR EACH clauses can be used in conjunction with the TRIGGERS, TERMINATES and INTERRUPTS statements.

The arrival of an INPUT may also cause an EVENT or set the value of a CONDITION.

CAUSES (CSS) _____;
(list of event names)

MAKES (MAK) _____TRUE (T);
(list of condition names)

MAKES (MAK) _____FALSE (F);
(list of condition names)

The DEPENDING ON and FOR EACH clauses may be used in conjunction with the CAUSES and MAKES statements.

An INPUT may or may not be involved in any system dynamics relationships.

3.4.6 Project-Management Statements for INPUT

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2

3.4.7 System-Properties Statements for INPUTS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.5 OUTPUT Section

OUTPUTS are information that is produced (GENERATED) by the target system (PROCESSES within the system) and that goes to (are RECEIVED BY) INTERFACES.

OUTPUT (OUT) _____;
(list of output names)

The name of the OUTPUT can be considered as the name attached to either or the collection of data values or the physical medium on which the data values are recorded, i.e., the carrier of the data values or to both.

3.5.1 System-Flow Statements for OUTPUTS

The names of the PROCESSES producing the OUTPUT are given in the GENERATED statement.

GENERATED BY (GEND) _____
(list of process names)

To specify the manner in which OUTPUTS are generated more precisely, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the GENERATED BY statement.

DEPENDING ON _____
(list of element/condition-names)

FOR EACH _____;
(list of
entity/input/output/group/element/set-names)

The INTERFACES which accept the OUTPUT are given in the RECEIVED BY statement:

RECEIVED BY (RCVD) _____
(list of interface names)

To specify the manner in which OUTPUTS are received more precisely, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the RECEIVED BY statement.

DEPENDING ON _____
(list of element/condition-names)

FOR EACH _____;
(list of
entity/input/output/group/element/set-names)

These statements refer only to the logical collection of data elements values, and provide a way of stating what PROCESSES must produce the OUTPUT and where it must be transmitted to.

All operations on the data element values must be specified separately in the definition of the PROCESS.

Every OUTPUT should be GENERATED by at least one PROCESS and RECEIVED by at least one INTERFACE.

3.5.2 System-Structure Statements for OUTPUTS

An OUTPUT may be part of one, and only one, larger OUTPUT, and it may have any number of subparts that are also OUTPUTS.

PART _____;
(name of output)

SUBPARTS (SUBP) _____;
(list of output names)

These statements allow definitional structures (grouping OUTPUTS together to call them a single name) and high level data structures to be specified. The lowest level of OUTPUTS normally will be used for physical documents, messages, cards, etc., that flow out of the system.

To describe a collection of OUTPUT occurrences (SETS of OUTPUTS) the CONTAINED statement may be used to relate OUTPUTS to SETS.

CONTAINED (CNTD) _____;
(list of set names)

This SET can then be used in further statement of requirements. This might be used, for example, to describe a batch of outputs that are to be produced as a unit.

An OUTPUT can be contained in any number of SETS.

3.5.3 Data-Structure Statements for OUTPUTS

The data (GROUPS and ELEMENTS) whose values appear on an OUTPUT are defined via the CONSISTS statement. Each data name used in the statement can be optionally preceded by a SYSTEM-PARAMETER to define the number of occurrences of the data value that may appear on the OUTPUT. The CONSISTS statement only specifies the data on the OUTPUT and implies nothing about format.

CONSISTS (CSTS) _____;
(list of group and element names,
each name optionally preceded by
a system parameter)

A complete problem should have all OUTPUTS that do not have SUBPARTS statements broken down to GROUPS and ELEMENTS.

The CLASSIFICATION of an OUTPUT may be specified with the CLASSIFICATION statment:

CLASSIFICATION _____;
(list of classification names, each optionally followed by a level number)

Any PROCESSES or PROCESSORS that use the OUTPUT must have SECURITY-ACCESS-RIGHTS that match the classification of the OUTPUT.

3.5.4 Data-Derivation Statements for OUTPUTS

The DERIVED statement specifies those PROCESSES that derive some information presented on the OUTPUT.

DERIVED (DRVD) _____;
(list of process names)

This statement implies that at least one piece of data (GROUP or ELEMENT) on the OUTPUT is DERIVED.

To specify more precisely how the OUTPUT is derived, the USING, DEPENDING ON, and FOR EACH clauses may be used in conjunction with the DERIVED statement.

DERIVED BY (DRVD) _____
(list of process names)

USING _____
(list of input, set, entity, group and element names)

DEPENDING ON _____
(list of element, condition-names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

3.5.5 System-Dynamics Statements for OUTPUTS

More than one individual instance of an OUTPUT may occur over some period of time. The number of instances of the OUTPUT that occur over time is stated through the HAPPENS statement:

HAPPENS (HAP) _____
(system-parameter)

TIMES-PER (TIMP) _____;
(interval name)

EVERY _____ _____;
(system-parameter) (interval name)

[WITHIN] _____ _____AFTER_____;
(system-parameter) (interval name) (event)

Every OUTPUT should have a HAPPENS statement.

3.5.6 Project-Management Statements for OUTPUTS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.5.7 System-Property Statements for OUTPUTS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.6 ENTITY Section

An ENTITY is a collection of information manipulated (USED, DERIVED and UPDATED) by the target system. An ENTITY differs from an INPUT or OUTPUT in that it is information maintained entirely internal to the system and can never cross the system boundaries (i.e., be GENERATED or RECEIVED).

INPUTS, OUTPUTS and ENTITIES are similar constructs, though only ENTITIES can be logically connected through RELATIONS.

ENTITY (ENT) _____;
(list of entity names)

In many applications, the usage of ENTITIES is synonymous with logical records. For example, if an employee payroll processing system were being designed, the information needed about salaried and hourly employees might be stored on records which would be defined as ENTITIES.

3.6.1 System Structure

To describe a collection of ENTITY occurrences (sometimes also called a file) the CONTAINED statement may be used to relate ENTITIES to SETS.

CONTAINED (ONTO) _____;
(list of set names)

This SET can then be used in further statement of requirements. This might be used, for example, to describe a file of employee records which are to be treated as a unit for processing.

3.6.2 Data-Structure Statements for ENTITIES

The data (GROUPS and ELEMENTS) whose values appear in an ENTITY are defined via the CONSISTS statement. Each data name used in the statement can be optionally preceded by a SYSTEM-PARAMETER to define the number of occurrences of the data value that may appear on the ENTITY.

The CONSISTS statement only specifies the data on the ENTITY and implies nothing about its format.

CONSISTS (CSTS) _____;
(list of group and element names, each name optionally preceded by a system parameter)

A complete problem statement should have all ENTITIES broken down to GROUPS and ELEMENTS.

To specify that each ENTITY occurrence may be uniquely identified by one or more keys, the IDENTIFIED statement is used.

IDENTIFIED (IDD) _____;
(list of group and element names)

The RELATED statement specifies a logical connection between two ENTITIES.

RELATED (REL) _____
(name of entity)

VIA _____;
(name of relation)

This implies that given one of the two ENTITIES, information from the other can be found.

3.6.3 Data-Derivation Statements for ENTITIES

The USED statement specifies those PROCESSES which use the information available in the ENTITY.

USED _____;
(list of process names)

This statement implies that at least one piece of data (GROUP or ELEMENT) in the ENTITY is being USED.

To specify the manner in which the ENTITY is USED more precisely, the DERIVE or UPDATE clause may be used in conjunction with the USED statment.

USED BY _____
(list of process names)

TO DERIVE (DRV) _____;
(list of element, group, entity set and output names)

or USED by a PROCESS to UPDATE data:

USED BY _____
(list of process names)

TO UPDATE (UPD) _____;
(list of element, group, entity and set names)

The DERIVED statement specifies those PROCESSES which derive some information presented in the ENTITY.

DERIVED (DRVD) _____:
(list of process names)

This statement implies that at least one piece of data (GROUP or ELEMENT) in the ENTITY is DERIVED. To specify the manner in which the ENTITY is derived more precisely, the USING, DEPENDING ON, and FOR EACH clauses may be used in conjunction with the DERIVED statement.

DERIVED BY (DRVD) _____
(list of process names)

USING _____:
(list of element, group, entity, set and input names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____:
(list of entity, input, output, group, element, set-name)

The UPDATED statement specifies those PROCESSES that modified some information presented in the ENTITY.

UPDATED (UPDD) _____:
(list of process names)

This statement implies that at least one piece of data (GROUP or ELEMENT) in the ENTITY is UPDATED.

To specify more precisely the manner in which the ENTITY is updated, the USING, DEPENDING ON, and FOR EACH clauses may be used in conjunction with the UPDATED statement.

UPDATED BY (UPDD) _____
(list of process names)

USING _____:
(list of element, group, entity, set or input names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____:
(list of entity, input, output, group, element, set-name)

Every ENTITY defined should be USED, DERIVED or UPDATED by at

least one PROCESS.

The CLASSIFICATION of an ENTITY may be specified with the CLASSIFICATION statement:

CLASSIFICATION _____;
(list of classification names, each optionally followed by a level number)

Any PROCESSES or PROCESSORS that use the ENTITY must have SECURITY-ACCESS-RIGHTS that match the classification of the ENTITY.

3.6.4 System-Size Statements for ENTITIES

The CARDINALITY statement specifies the maximum number of occurrences of a particular ENTITY in the target system at any time.

CARDINALITY (CARD) _____;
(system parameter)

Every ENTITY should have a CARDINALITY.

3.6.5 System-Dynamics Statements for ENTITIES

The VOLATILITY statement specifies the manner in which an ENTITY changes over time. Since there are many different ways in which an ENTITY may be changed, this information is entered via a comment entry. The type of information specified in this statement might be the number of times a particular ENTITY occurrence would be updated in a given time interval, how often ENTITY occurrences would be deleted, and often created, etc.

VOLATILITY (VOL);

_____ _____
_____ _____ ;
(comment entry)

Every ENTITY should have a VOLATILITY.

3.6.6 Project-Management Statements for ENTITIES

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.6.7 System-Properties Statements for ENTITIES

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.7 SET Section

A SET is a collection of one or more ocurrences of objects that contain or carry data values. A SET may represent a collection of ENTITIES, INPUTS, or OUTPUTS, but not a combination of these object types. That is, a SET cannot consist of both INPUTS and OUTPUTS.

SET _____;
(list of set names)

Where ENTITIES may be thought of as logical records, a SET may be thought of as a logical file. In any case, a SET should be used according to the algebraic sense of the word "set."

3.7.1 System-Flow Statements for SETS

The RESPONSIBLE-INTERFACE statement specifies those INTERFACES that have the responsibility of maintaining the information in the SET.

RESPONSIBLE-INTERFACE (RINT) _____;
(list of interface names)

Every SET should have at least one responsible INTERFACE.

3.7.2 System-Structure Statements for SETS

The SUBSETS and SUBSET statements specify the manner in which a particular SET is related (in the algebraic sense, again) to other SETS in the target system.

A SET can be a SUBSET of a larger (or equivalent size) SET:

SUBSET (SS) _____;
(list of set names)

A SET can also have a number of SUBSETS:

SUBSETS (SSS) _____;
(list of set names)

For example, a data base may be defined to describe all the information maintained by the target system. The data base may be defined to be a SET. Smaller collections of data in the data base such as files, etc., would then be defined as SUBSETS of the data base.

The SUBSETTING-CRITERIA statement specifies what data determines how a SET is to be subsetted.

SUBSETTING-CRITERIA (SSCA) _____;
(list of subsetting-criterion, element, and group names)

If a SET has SUBSETS, its SUBSETTING-CRITERIA should be defined also.

3.7.3 Data-Structure Statements for SETS

The CONSISTS statement specifies the data contained in the SET and, optionally, the number of occurrences of this data in the SET.

CONSISTS (CSTS) _____;
(list of entity, input or output names, optionally preceded by system-parameters)

Every SET should CONSIST of at least one ENTITY, INPUT, or OUTPUT.

3.7.4 Data-Derivation Statements for SETS

The USED statement specifies those PROCESSES which use the information available in the SET.

USED _____;
(list of process names)

This implies that some data within the SET is being USED.

To specify the manner in which the SET is USED more precisely, the DERIVE or UPDATE clause may be in conjunction with the USED statement.

USED BY _____
(list of process names)

TO DERIVE (DRV) _____;
(list of element, group, entity, set and output names)

or USED by a PROCESS to UPDATE data:

USED BY _____
(list of process names)

TO UPDATE (UPD) _____;

(list of element, group, entity,
and set names)

The DERIVED statement specifies those PROCESSES which derived some information presented in the SET.

DERIVED (DRVD)----------;
(list of process names)

This statement implies that at least one piece of data (ENTITY or OUTPUT) in the SET is DERIVED. To specify the manner in which the ENTITY or OUTPUT is derived more precisely, the USING, DEPENDING ON, and FOR EACH clauses may be used in conjunction with the DERIVED statement.

DERIVED BY (DRVD) _____
(list of process names)

USING----------;
(list of element, group, entity,
set and input names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group,
element, set-name)

The UPDATED statement specifies those PROCESSES that may modify information in the SET.

UPDATED (UPDD) _____;
(list of process names)

This statement implies that at least one piece of data (ENTITY) in the SET is UPDATED.

To specify more precisely the manner in which the SET is updated, the USING, DEPENDING ON, and FOR EACH clauses may be used in conjunction with the UPDATED statement.

UPDATED BY (UPDD) _____
(list of process names)

USING----------;
(list of element,group, entity,
set or input names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group,

element, set-name)

Every SET defined should be USED, DERIVED or UPDATED by at least one PROCESS.

The DERIVATION statement should be used to specify the rules for deriving occurrences of data in the SET. Since there are many different ways in which this data may be derived, this information is presented via a comment entry. The type of information specified in this statemnt might be what value a particular ELEMENT in an ENTITY must have to be entered into a SET, etc.

DERIVATION (DRVN);

---------- ;
(comment entry)

Every SET should have DERIVATION specified.

The CLASSIFICATION of a SET may be specified with the CLASSIFICATION statement:

CLASSIFICATION----------;
(list of classification names,
each optionally followed by
a level number)

Any PROCESSES or PROCESSORS that use the SET must have SECURITY-ACCESS-RIGHTS that match the classification of the SET.

3.7.5 System-Size Statements for SETS

The CARDINALITY statement specifies the maximum number of occurrences of data objects in the SET at any one time.

CARDINALITY (CARD) ----------;
(system parameter)

Every SET should have a CARDINALITY.

3.7.6 System-Dynamics Statements for SETS

The VOLATILITY-SET and VOLATILITY-MEMBER statements specify how a SET changes over time. Since there are many different ways in which a SET may be changed, this information is presented via a comment entry.

The VOLATILITY-SET statement specifies the manner in which the entire set changes over time. The type of information specified in this statement might be the number of times members are added to the SET, members are updated, etc.

VOLATILITY-SET (VOLS);

--- ;
(comment entry)

The VOLATILITY-MEMBER statement specifies how the members of the SET change over time. The type of information specified in this statement might be the number of additions to the SET of a particular ENTITY type, the number deleted, etc.

VOLATILITY-MEMBER (VOLM);

--- ;
(comment entry)

Every SET should have VOLATILITY-SET and VOLATILITY-MEMBER statements given for them.

3.7.7 Project-Management Statements for SETS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.7.8 System-Properties Statements for SETS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.8 RELATION Section

A RELATION is a named logical connection between two ENTITIES perceived by the Problem Definer. Any URL name may be used; the most meaningful name to the Problem Definer should be one which denotes the connected ENTITIES.

RELATION _____;
(list of relation names)

If a system were being described that consisted of ENTITIES for women and ENTITIES for men, a possible RELATION to connect these ENTITIES might be "spouse."

3.8.1 Data-Structure Statements for RELATIONS

A BETWEEN statement specifies the names of the ENTITIES that the RELATION connects and the direction of the connection. The direction is determined by the order of the ENTITY names in the statement: from the left (first) ENTITY to the right (second) ENTITY. The first ENTITY can be considered the owner of the RELATION and the second ENTITY the member of the RELATION.

BETWEEN _____
(name of entity)

AND _____;
(name of entity)

Example: BETWEEN DEPARTMENT-RECORD AND HOURLY-EMPLOYEE-RECORD:

The RELATION, DEPARTMENT-TO-HOURLY-EMPLOYEE, denotes a logical connection between two ENTITIES, DEPARTMENT-RECORD and HOURLY-EMPLOYEE-RECORD. The direction is from DEPARTMENT-RECORD to HOURLY-EMPLOYEE-RECORD. The DEPARTMENT-RECORD is the owner and HOURLY-EMPLOYMENT-RECORD the member of the RELATION.

Only one BETWEEN statement can be given for a particular RELATION, but each RELATION should have a BETWEEN statement given for it.

The ASSOCIATED-DATA statement specifies those GROUPS and ELEMENTS that contain information specifically about the RELATION and are not necessarily CONTAINED in either ENTITY.

ASSOCIATED-DATA IS _____;
(list of element and group names)

3.8.2 Data-Derivation Statements for RELATIONS

A MAINTAINED BY statement designates those PROCESSES which add, delete or modify the connection occurrences between the ENTITIES that are connected by this RELATION.

MAINTAINED BY _____;
(list of process names)

To specify more precisely the manner in which the RELATION is maintained, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the MAINTAINED BY statement.

MAINTAINED BY _____
(list of process names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

A RELATION can be MAINTAINED by several PROCESSES, and every RELATION should be MAINTAINED by at least one PROCESS.

The DERIVATION statement should be used to specify the rules for deriving occurrences of the RELATION between the ENTITIES . Since there are many different ways in which this data may be derived, this information is presented via a comment entry. The type of information specified in this statement might be what are the restrictions in relating two ENTITIES, which PROCESSES may forma the relation, etc.

DERIVATION (DRVN);

_____
_____ ;
(comment entry)

Every RELATION should have a DERIVATION specified.

3.8.3 System-Size Statements for RELATIONS

A CONNECTIVITY statement specifies the number of ENTITY occurrences of the first (right) ENTITY that are related to a number of ENTITY occurrences of the second (left) ENTITY.

CONNECTIVITY IS _____
(system-parameter)

TO _____;
(system-parameter)

If a particular ENTITY occurrence may be related to only one other ENTITY occurrence, the CONNECTIVITY is 1 to 1. If a particular ENTITY occurrence may be related to one or more ENTITY occurrences the CONNECTIVITY is one to many. The right and left SYSTEM-PARAMETERS in the CONNECTIVITY are intended to correspond to the right and left ENTITIES given in the BETWEEN statement.

Every RELATION should have one, and only one, CONNECTIVITY.

A CARDINALITY statement specifies the maximum number of connection occurrences for this RELATION.

CARDINALITY IS _____;
(system-parameter)

Every RELATION should have one, and only one, CARDINALITY.

3.8.4 Project-Management Statements for RELATIONS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statemnt is given in section 3.2.

3.8.5 System-Properties Statements for RELATIONS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this section. Description and syntax of these statements are given in section 3.2.

3.8.6 Example of a Complete RELATION Section

```
RELATION department-to-hourly-employee;

     ASSOCIATED-DATA is last-department-change;
     ATTRIBUTE IS frequency-of-use: high;
     BETWEEN department-record AND hourly-employee-record;
     CARDINALITY IS number-of-hourly-employees;
     CONNECTIVITY IS 1 TO max-department-employement;
     DERIVATION;
        new-employee-processing adds connections while
        terminating-employee-processing deletes connections;
     DESCRIPTION;
        this relation connects an hourly-employee-record for
        each employee in a department to the department-record
        for that department;
     KEYWORD department-information;
     MAINTAINED BY new-employee-processing AND
        terminating-employee-processing /* USING
        department AND employee-identification-number */;
```

```
RESPONSIBLE-PROBLEM-DEFINER john-proctor;
SECURITY department-heads, department-secretaries;
SEE-MEMO company-organization-chart;
SOURCE   employee-application-form,
         employee-termination-form,
         department-employee-list;
SYNONYM dept-to-emp, d-e;
```

3.3 GROUP and ELEMENT Sections

An ELEMENT is the lowest level data object that can be defined to describe data. Because of this property, an ELEMENT has one or more possible data values associated with it, whether it be alphabetic, numeric or otherwise. In many instances an ELEMENT may be thought of synonymously with "field" or "item."

ELEMENT (ELE) _____;
(list of element names)

A GROUP represents a collection of ELEMENTS and/or GROUPS. The use of GROUPS is definitional which means that referencing a particular GROUP by its name is equivalent to referencing the individual ELEMENTS which the GROUP consists of.

GROUP (GR) _____;
(list of group names)

GROUPS can be broken down into smaller GROUPS and ELEMENTS, but ELEMENTS cannot be subdivided. ELEMENTS may take on values where GROUPS may not. The value of a GROUP is defined to be equivalent to the individual values of the ELEMENTS within the GROUP.

3.3.1 Data-Structure Statements for GROUPS and ELEMENTS

The CONTAINED statement is used to relate the data structure relationships of GROUPS and ELEMENTS to ENTITIES, INPUTS and OUTPUTS. Data is most often thought to be part of some large unit of data such as a logical record, input form, or output report, which can be represented by the ENTITY, INPUT and OUTPUT, respectively.

CONTAINED (CNTD) _____;
(list of group, entity,
input and output names)

GROUPS and ELEMENTS may be defined to be CONTAINED in some larger GROUP.

The CONSISTS statement is used to specify those lower level GROUPS and ELEMENTS a GROUP may consist of. By definition of "ELEMENT," an ELEMENT cannot CONSIST of any other data objects. The CONSISTS statement only specifies the data in the GROUP and implies nothing about its format.

CONSISTS (CSTS) _____;
(list of group and element names, optionally
preceded by system parameters)

A complete problem statement should have all GROUPS broken down to smaller GROUPS and/or ELEMENTS.

The ASSOCIATED statement specifies those RELATIONS that the GROUPS or ELEMENTS are associated with. This implies that the information in the GROUP or ELEMENT is in neither of the ENTITIES the RELATION is BETWEEN.

ASSOCIATED (ASSO) _____;
(list of relation names)

The IDENTIFIES statement specifies those ENTITIES for which the GROUP or ELEMENT is used as an identification key. This implies that the possible values of the GROUP or ELEMENT are all unique. For example, the ELEMENT which represents social security number in an employee record might be used as an identifier.

IDENTIFIES (IDS) _____;
(list of entity names)

A GROUP or ELEMENT may identify any number of ENTITIES.

3.9.2 System-Structure Statements for GROUPS and ELEMENTS

The SUBSETTING-CRITERION statement specifies those SETS which are subsetted based on the data values in the GROUP or ELEMENT.

SUBSETTING-CRITERION (SSCN) _____;
(list of set names)

3.9.3 Data-Derivation Statements for GROUPS and ELEMENTS

The USED statement specifies those PROCESSES which use the information in the GROUP or ELEMENT.

USED _____;
(list of process names)

This statement implies (in the case of a GROUP) that at least one piece of data in the GROUP is being USED.

To specify the manner in which the GROUP is USED more precisely, the DERIVE or UPDATE clause may be used in conjunction with the USED statement.

USED BY _____
(list of process names)

TO DERIVE (DRV) _____;
(list of element, group, entity, set and output names)

or USED by a PROCESS to UPDATE data:

USED BY _____;
(list of process names)

TO UPDATE (UPD) _____;
(list of element, group, entity, and set names)

The DERIVED statement specifies those PROCESSES that derived some information presented in the GROUP or ELEMENT.

DERIVED (DRVD) _____;
(list of process names)

This statement implies (in the case of a GROUP) that at least one piece of data (GROUP or ELEMENT) in the GROUP is DERIVED. To specify more precisely the manner in which the GROUP or ELEMENT is derived, the USING, DEPENDING ON, and FOR EACH clauses may be used in conjunction with the DERIVED statement.

DERIVED BY (DRVD) _____
(list of process names)

USING _____;
(list of element, group, entity, set and input names)

DEPENDING ON _____ _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

The UPDATED statement specifies those PROCESSES that modify some information presented in the GROUP or ELEMENT.

UPDATED (UPDD) _____;
(list of process names)

This statement implies (in the case of a GROUP) that at least one piece of data (GROUP or ELEMENT) in the GROUP is UPDATED.

To specify more precisely the manner in which the GROUP or ELEMENT is updated, the USING, DEPENDING ON, and FOR EACH clauses may be used in conjunction with the UPDATED statement.

UPDATED BY (UPDD) _____
(list of process names)

USING _____;
(list of element, group, entity, set or input names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

Every GROUP and ELEMENT defined should be USED, DERIVED or UPDATED by at least one PROCESS.

The CLASSIFICATION of a GROUP or ELEMENT may be specified with the CLASSIFICATION statement:

CLASSIFICATION _____;
(list of classification names, each optionally followed by a level number)

Any PROCESSES or PROCESSORS that use the GROUP or ELEMENT must have SECURITY-ACCESS-RIGHTS that match the classification of the GROUP or ELEMENT.

3.9.4 System-Size Statements for ELEMENTS

The VALUE statement is used to define numeric values an ELEMENT may have. A GROUP cannot have a VALUE directly associated with it. The VALUE statement may only specify numeric values and does not imply anything about storage format, etc. The ATTRIBUTES and DESCRIPTION statement should be used to present this type of information as well as to specify character values.

VALUE (VAL) _____;
(integer value)

Only positive integer values may be specified. Decimal numbers, negative numbers, etc. are not acceptable.

A range of values may also be specified.

VALUES (VAL) _____ THRU _____;
(minimum value) (maximum value)

Again, the values must be positive integers. POSINF and NEGINF may be used to represent positive and negative infinity, respectively.

Only one VALUE statement, of either of the forms, may be given to describe a particular ELEMENT.

3.9.5 Project-Management Statements for GROUPS and ELEMENTS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.9.6 System-Properties Statements for GROUPS and ELEMENTS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.10 PROCESS Section

The PROCESS is used to define the function, or functions of the target system. At the highest level, the function of the target system may be defined as a single PROCESS. This PROCESS, could in turn, be broken down into more detailed PROCESSES. It is the task of the PROCESS to reference and manipulate data in the target system.

PROCESS (PRC) _____;
(list of process names)

3.10.1 System-Flow Statements for PROCESSES

The RECEIVES statement is used to specify that the PROCESS accepts information (INPUTS) from outside the target system.

RECEIVES (RCVS) _____;
(list of input names)

This statement only specifies that the INPUTS are accepted by the PROCESS and does not imply that the information in the INPUTS are USED or how it is USED by the PROCESS.

To specify more precisely the manner in which the INPUTS are received, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the RECEIVES statement.

RECEIVES _____
(list of input names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

The GENERATES statement is used to specify that the PROCESS produces information (OUTPUTS) for use outside the target system.

GENERATES (GENS) _____;
(list of output names)

This statement only specifies that the OUTPUTS are distributed by the PROCESS, and does not imply that the information in the OUTPUTS is DERIVED by the PROCESS.

To specify more precisely the manner in which the OUTPUTS are generated, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the GENERATES statement.

GENERATES _____
(list of output names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

These statements imply that some physical processing or translation may be necessary. The RECEIVES statement means that the physical media containing data must be accommodated. Similarly, the GENERATES statement means that data must be recorded in whatever medium has been chosen.

3.10.2 System-Structure Statements for PROCESSES

A PROCESS may be part of one, and only one, larger PROCESS, and it may have any number of subparts that are also PROCESSES.

PART _____;
(process name)

SUBPARTS (SUBP) _____:
(list of process names)

These statements permit organization functions and programming structures to be defined for the problem statement.

The UTILIZED and UTILIZES statements are used to specify that a PROCESS represents a function used by several other PROCESSES. Definition of UTILIZED implies that the PROCESS is common to more than one other PROCESS. If not, the PROCESS should be defined as a SUBPART.

UTILIZED (UTLD) _____:
(list of process names)

UTILIZES (UTIS) _____:
(list of process names)

A given PROCESS may have any number of SUBPARTS and UTILIZE any number of other PROCESSES. A PROCESS may be a SUBPART of only one other PROCESS, but be UTILIZED by any number of PROCESSES.

To specify more precisely the manner in which the PROCESSES are utilized, the DPENDING ON and FOR EACH clauses may be used in conjunction with UTILIZES statement.

UTILIZES _____
(list of process names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

3.10.3 Data-Derivation Statements for PROCESSES

The USES statement specifies those SETS, INPUTS, ENTITIES, GROUPS and ELEMENTS from which some information is taken and used by the PROCESS to perform its designated function.

USES _____;
(list of set, input, entity, group and element names)

In the case where SET, INPUT, ENTITY or GROUP names are given, this statement implies that at least one ELEMENT within these are USED by the PROCESS.

To specify the manner in which the PROCESS USES the data more precisely, the DERIVE or UPDATE clause may be used in conjunction with the USES statement.

USES _____
(list of set, input, entity, group and element names)

TO DERIVE _____;
(list of set, output, entity, group and element names)

USES _____
(list of set, input, entity, group and element names)

TO UPDATE _____;
(list of set, entity, group and element names)

The DERIVES statement specifies those SETS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS for which some information is derived by the PROCESS to perform its designated function.

DERIVES (DRVS) _____;
(list of set, output, entity, group and element names)

In the case where SET, OUTPUT, ENTITY and GROUP names are given, this statement implies that at least one ELEMENT within these are DERIVED by the PROCESS.

To specify the manner in which the PROCESS DERIVES the data more precisely, the USING, DEPENDING ON and FOR EACH clauses may be used in conjunction with the DERIVES statement.

DERIVES (DRVS) _____
(list of set, output, entity,
group and element names)

USING _____;
(list of set, input, entity,
group and element names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group,
element, set-names)

The UPDATES statement specifies those SETS, ENTITIES, GROUPS and ELEMENTS for which some information is updated by the PROCESS in performing its designated function.

UPDATES (UPDS) _____;
(list of set, entity,
group and element names)

In the case where SET, ENTITY and GROUP names are given, this statement implies that at least one ELEMENT within these are UPDATED by the PROCESS.

To specify the manner in which the PROCESS UPDATES the data more precisely, the USING, DEPENDING ON and FOR EACH clauses may be used in conjunction with the UPDATES statement.

UPDATES (UPDS) _____
(list of set, entity,
group and element names)

USING _____;
(list of set, input, entity,
group and element names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group,
element, set-names)

The MAINTAINS statement specifies those RELATIONS or SUBSETTING-CRITERION which are maintained by the PROCESS. Maintenance of RELATIONS normally involves addition and deletion of connections between ENTITIES whereas maintenance of SUBSETTING-CRITERION deals with placement of ENTITIES, INPUTS and OUTPUTS in proper SETS according to the values of the ELEMENTS and GROUPS contained within those designated as SUBSETTING-CRITERION names.

MAINTAINS (MTNS) _____;
(list of relation and subsetting criterion names)

To specify more precisely the manner in which the PROCESS maintains a RELATION or SUBSETTING-CRITERION, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the MAINTAINS statement.

MAINTAINS _____
(list of relation and subsetting criterion names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____:
(list of entity, input, output, group, element, set-names)

Every PROCESS should be defined to interact with data in some manner (DERIVES, USES, UPDATES or MAINTAINS).

The PROCEDURE statement is used to specify an algorithm of the function of the PROCESS. The PROCEDURE statement is a comment entry statement thus allowing any form of procedure specification to be given such as decision tables, actual program code, narrative format, etc.

PROCEDURE (PRCD);

_____
_____ ;
(comment entry)

Every PROCESS that does not have SUBPARTS or does not UTILIZE any other PROCESSES should have a PROCEDURE statement that specifies, in sufficient detail for implementation, the rules for carrying out its function.

The SECURITY-ACCESS-RIGHTS of a PROCESS may be specified with the SECURITY-ACCESS-RIGHTS statement:

SECURITY-ACCESS-RIGHTS _____;
(list of classification names, each optionally followed by a level number)

A PROCESS that uses, derives or updates data must have SECURITY-ACCESS-RIGHTS that match the classification of the data.

3.10.4 System-Size Statements for PROCESSES

The HAPPENS statement is used to specify the frequency of a PROCESS in a given time interval.

HAPPENS (HAP) _____TIMES-PER (TIMP) _____;
(system parameter) (interval name)

HAPPENS EVERY _____ _____;
(system-parameter) (interval name)

HAPPENS [WITHIN] _____ _____AFTER _____;
(system parameter) (interval name) (Event)

3.10.5 System-Dynamics Statements for PROCESSES

The TRIGGERED, TERMINATED and INTERRUPTED statements are used to specify those EVENTS, INPUTS, PROCESSES and CONDITIONS that affect the initialization of processing, or the halting of processing.

TRIGGERED BY (TRGD) _____;
(list of event, input and/or process names)

To specify more precisely the manner in which an EVENT, INPUT or PROCESS is triggered, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the TRIGGERED BY statement.

TRIGGERED BY _____
(list of event, input and/or process names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

TRIGGERED WHEN (TRGD) _____BECOMES TRUE (T);
(condition name)

TRIGGERED WHEN (TRGD) _____BECOMES FALSE (F);
(condition name)

TERMINATED BY (TRMD) _____;
(list of event, input and/or process names)

To specify more precisely the manner in which an EVENT, INPUT or PROCESS is terminated, the DEPENDING ON and FOR EACH clauses may be used in conjunction with the TERMINATED BY statement.

TERMINATED BY _____
(list of event, input and/or process names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

To specify more precisely the manner in which an EVENT, INPUT or PROCESS are interrupted, the DEPENDING ON and FOR EACH statements may be used in conjunction with the INTERRUPTED BY statement.

INTERRUPTED BY _____
(list of event, input and/or process names)

DEPENDING ON _____
(list of element, condition names)

FOR EACH _____;
(list of entity, input, output, group, element, set-names)

TERMINATED WHEN (TRMD) _____BECOMES TRUE (T);
(condition name)

TERMINATED WHEN (TRMD)----------BECOMES FALSE (F);
(condition name)

INTERRUPTED BY (INTD)----------;
(list of event, input and/or process names)

INTERRUPTED WHEN (INTD) _____BECOMES TRUE (T);
(condition name)

INTERRUPTED WHEN (INTD) _____BECOMES FALSE (F);
(condition name)

PROCESSES may also TRIGGER, TERMINATE and INTERRUPT other PROCESSES.

TRIGGERS (TRGS) _____;
(list of process names)

TERMINATES (TRMS) _____;
(list of process names)

INTERRUPTS (INTS) _____;
(list of process names)

PROCESSES may also generate EVENTS and set values of CONDITIONS. An EVENT may be generated either at the initiation of a PROCESS or when it finishes.

INCEPTION-CAUSES (INCC) _____;
(list of event names)

TERMINATION-CAUSES (TERC) _____;
(list of event names)

MAKES _____TRUE (T);
(list of condition names)

MAKES _____FALSE (F);
(list of condition names)

The INCEPTION-CAUSES, TERMINATION-CAUSES and MAKES statements allow the use of the optional clauses DEPENDING ON and FOR EACH.

A PROCESS may or may not be involved in any system dynamics relationships.

3.10.6 System-Architecture Statements for PROCESSES

The PERFORMED statement specifies the physical PROCESSOR (e.g., hardware or organizational unit) which performs the functions described by the PROCESS.

PERFORMED (PFMD) _____;
(name of processor)

Every PROCESS should be PERFORMED by some PROCESSOR.

The RESOURCE-USAGE statement indicates resource consumption associated with the PROCESS.

RESOURCE-USAGE (RU) _____FOR _____;
(system parameter) (name of resource-usage-parameter

3.10.7 Project-Management Statements for PROCESSES

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.10.8 System-Property Statements for PROCESSES

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.11 INTERVAL Section

An INTERVAL is used to define a segment of time. A week or day are simple examples of INTERVALS.

INTERVAL (INT) _____;
(list of interval names)

It is important to note that unless defined as a SYNONYM, WEEKS is not the same as WEEK. In most cases, it is desirable that both names represent the same interval.

3.11.1 System-Structure Statements for INTERVALS

The CONSISTS statement specifies the smaller INTERVALS that the INTERVAL can be broken down to.

CONSISTS (CSTS) _____;
(list of interval names, each optionally preceded by a system parameter)

The SYSTEM-PARAMETER should be specified to make the relationship between intervals meaningful. It makes little sense to say that a year consists of weeks without the quantitative property.

3.11.2 Project-Management Statements for INTERVALS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax for this statement are given in section 3.2.

3.11.3 System-Properties Statements for INTERVALS

The SYNONYM, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.12 CONDITION Section

A CONDITION designates some situation that the problem definer wants to identify because it influences the requirements for the system.

CONDITION (COND) _____;
(list of condition names)

3.12.1 System Structure Statements for CONDITIONS

The interdependencies among conditions and data will be declared with a DEPENDS statement in the following manner.

CONDITION _____
(list of condition names)

DEPENDS ON _____;
(list of element or condition names)

The first of the month may represent some CONDITION for which action of the target system would occur depending on the state of the CONDITION.

3.12.2 System-Dynamics Statements for CONDITIONS

The TRUE WHILE and FALSE WHILE statements specify those situations when the CONDITION is in the TRUE state, or in the FALSE state, respectively. This information is presented in a comment entry format.

TRUE WHILE;

_____
_____ ;
(comment entry)

FALSE WHILE;

_____
_____ ;
(comment entry)

Every CONDITION should have a TRUE WHILE or a FALSE WHILE statement.

A CONDITION can be set by a PROCESS, an EVENT or the arrival of an INPUT.

MADE TRUE BY _____;
(list of processes, events and/or inputs)

MADE FALSE BY _____;
(list of processes, events and/or inputs)

The change in state of a CONDITION may also affect the processing being performed, or may initiate new processing.

BECOMING (BECG) TRUE (T) TRIGGERS (TRGS) _____:
(list of process names)

BECOMING (BECG) FALSE (F) TRIGGERS (TRGS) _____:
(list of process names)

BECOMING (BECG) TRUE (T) TERMINATES (TRMS) _____:
(list of process names)

BECOMING (BECG) FALSE (F) TERMINATES (TRMS) _____:
(list of process names)

BECOMING (BECG) TRUE (T) INTERRUPTS (INTS) _____:
(list of process names)

BECOMING (BECG) FALSE (F) INTERRUPTS (INTS) _____:
(list of process names)

The change in state of a condition may cause an EVENT.

BECOMING (BECG) TRUE (T) CAUSES (CSS) _____:
(list of event names)

BECOMING (BECG) FALSE (F) CAUSES (CSS) _____:
(list of event names)

A CONDITION should interact in some way with at least one EVENT or PROCESS.

3.12.3 Project-Management Statements for CONDITIONS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.12.4 System-Properties Statements for CONDITIONS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.13 EVENT Section

An EVENT defines an occurrence of something within the system. The state of a CONDITION, initiation of a PROCESS, etc. may be defined as EVENTS.

EVENTS (EV) _____;
(list of event names)

An EVENT occurs at a given instant in time and is used in the problem statement to relate the things that go on in the system with time.

3.13.1 System-Dynamics Statements for EVENTS

An EVENT may be caused by a PROCESS (either on inception or on termination), a CONDITION, an INPUT or another EVENT.

CAUSED BY (CSD) _____;
(list of event and/or input names)

CAUSED WHEN (CSD) _____BECOMES TRUE (T);
(condition name)

CAUSED WHEN (CSD) _____BECOMES FALSE (F);
(condition name)

ON INCEPTION (INCP) _____;
(list of process names)

ON TERMINATION (TERM) _____;
(list of process names)

An EVENT may cause another EVENT or set the value of a CONDITION.

CAUSES (CSS) _____;
(list of event names)

MAKES (MAK) _____TRUE (T);
(list of condition names)

MAKES (MAK) _____FALSE (F);
(list of condition names)

An EVENT may affect processing, or initiate new procesing.

TRIGGERS (TRGS) _____;
(list of process names)

TERMINATES (TRMS) _____;
(list of process names)

INTERRUPTS (INTS) _____:
(list of process names)

The CAUSES BY, ON INCEPTION, ON TERMINATION, CAUSES, MAKES, TRIGGERS, TERMINATES and INTERRUPTS statements allow the use of the optional clauses DEPENDING ON and FOR EACH.

An EVENT should interact with at least one CONDITION or PROCESS.

The HAPPENS statement specifies the frequency of the EVENT in the target system for a given time interval.

HAPPENS (HAP) _____TIMES-PER (TIMP)_____;
(system parameter) (interval name)

HAPPENS EVERY _____ _____;
(system-parameter) (interval name)

HAPPENS [WITHIN] _____ _____AFTER
_____:
(system parameter) (interval name) (event)

Every EVENT should have one, and only one, HAPPENS statement.

3.13.2 Project-Management Statements for EVENTS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax of this statement are given in section 3.2.

3.13.3 System-Properties Statements for EVENTS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.14 PROCESSOR Section

A PROCESSOR is an object that can "perform" a PROCESS. That is, a PROCESSOR is an "agent," such as a computer system, organizational unit, or person, that physically acts to perform a PROCESS.

PROCESSOR (PRCR) _____:
(list of processor names)

3.14.1 System-Structure Statements for PROCESSORS

A PROCESSOR may be part of one, and only one, larger PROCESSOR, and it may have any number of subparts that are also PROCESSORS.

PART _____;
(processor name)

SUBPARTS (SUBP) _____;
(list of processor names)

3.14.2 Data-Derivation Statements for PROCESSORS

In the target system, PROCESSOR may have the right to access information of certain classifications and categories.

SECURITY-ACCESS-RIGHT (SAR) _____;
(list of classification names optionally followed by classification levels)

3.14.3 System-Architecture Statements for PROCESSORS

A PROCESSOR may CONSUME RESOURCES, such as CPU-time, elapsed time, or memory.

CONSUMES (CNSS) _____
(name of resource)

RATE _____
(system-parameter)

PER _____;
(name of resource-usage-parameter)

A PROCESSOR is the object, group or person that performs the functions specified by one or more PROCESSES.

PERFORMS (PFMS) _____;
(list of process names)

3.14.4 Project-Management Statements for PROCESSORS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax for this statement are given in section 3.2.

3.14.5 System-Property Statements for PROCESSORS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax for these statements are given in section 3.2.

3.15 RESOURCE Section

A RESOURCE is something that the physical elements of the target system consume in order to carry out information processing functions.

RESOURCE (RSC) _____:
(name of resource)

3.15.1 System-Architecture Statements for RESOURCES

A RESOURCE may be consumed, or used up, by a PROCESSOR.

CONSUMED (CNSD) _____
(list of processor names)

RATE _____PER_____;
(system parameter) (name of resource-usage-parameter)

Resource usage must be measured in some unit, such as milliseconds or feet.

MEASURED (MSRD) _____;
(name of unit)

3.15.2 Project-Management Statements for RESOURCES

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax for this statement are given in section 3.2.

3.15.3 System Property Statements for RESOURCES

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax for these statements are given in section 3.2.

3.16 RESOURCE-USAGE-PARAMETER Section

A RESOURCE-USAGE-PARAMETER is an object that defines a measure of the RESOURCE usage for a PROCESS. It is used to express resource consumption of a PROCESSOR performing a PROCESS independent of what PROCESSOR performs it.

RESOURCE-USAGE-PARAMETER (RUP) _____:
(name of resource-usage-parameter)

3.16.1 System-Architecture Statements for RESOURCE-USAGE-PARAMETERS

A particular value for RESOURCE usage may be associated with a given PROCESS.

RESOURCE-USAGE-PARAMETER-VALUE (RUPV)----------
(system parameter)

FOR _____;
(name of process)

3.16.2 Project-Management Statements for RESOURCE-USAGE-PARAMETERS

The RESPONSIBLE-PROBLEM-DEFINER statment may be used in this section. Description and syntax for this statement are given in section 3.2.

3.16.3 System-Property Statements for RESOURCE-USAGE-PARAMETERS

The SYNONYMs, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this section. Description and syntax for these statements are given in section 3.2.

3.17 UNIT Section

A UNIT is used to measure RESOURCES. For example, possible UNITS would include inches and kilowatt hours.

UNIT _____;
(name of unit)

3.17.1 System-Architecture Statements for UNITS

A UNIT must be associated with the RESOURCES it is used to measure.

MEASURES (MSRS) _____;
(list of resource names)

3.17.2 Project-Management Statements for UNITS

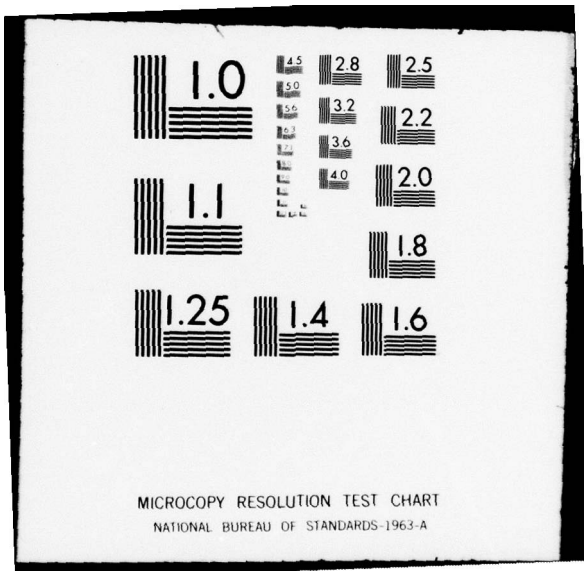
The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this section. Description and syntax for this statement are given in section 3.2.

AD-A058 629 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART I. (DESCRI--ETC(U)
JUL 78 F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-130-VOL-1 NL

3 OF 3
AD
A058629

END
DATE
FILMED
11-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

3.17.3 System-Property Statements for UNITS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTE, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax for these statements are given in section 3.2.

3.18 PROBLEM-DEFINER Section

The PROBLEM-DEFINER is that person responsible for one or more sections of the URL Problem Statement. In most cases, this is the person who originally wrote those URL statements.

PROBLEM-DEFINER (PD) _____;
(list of problem definer names)

3.18.1 Project-Management Statements for PROBLEM-DEFINER

The RESPONSIBLE statement is used to specify those URL sections for which the PROBLEM-DEFINER is responsible.

RESPONSIBLE (RESP) _____;
(list of names)

A PROBLEM-DEFINER cannot be RESPONSIBLE for other PROBLEM-DEFINERS.

A PROBLEM-DEFINER should be RESPONSIBLE for at least one name.

The MAILBOX statement specifies an address for the PROBLEM-DEFINER to which comments or questions concerning the problem statement can be sent.

MAILBOX (BOX) _____;
(name of mailbox)

A PROBLEM-DEFINER may have only one MAILBOX.

3.18.2 System-Properties Statements for PROBLEM-DEFINERS

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

3.19 MEMO Section

A MEMO is a narrative description which applies to more than one name in the problem statement.

MEMO _____;
(memo name)

The text of the MEMO should be put in the DESCRIPTION statement.

3.19.1 Project-Management Statements for MEMOS

The RESPONSIBLE-PROBLEM-DEFINER statement may be used is this Section. Description and syntax of this statement are given in section 3.2.

3.19.2 System-Properties Statements for MEMOS

The SYNONYMS, DESCRIPTION, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

The APPLIES statement specities those URL names to which the MEMO pertains.

APPLIES (APP) _____;
(list of names)

A MEMO cannot APPLY to another MEMO name.

A MEMO should APPLY to at least two names. Otherwise, the information could be presented in the DESCRIPTION statement for the name.

3.20 The DEFINE Section

The DEFINE section is used to specify information about special types of names that do not have their own URL sections.

The format of the DEFINE section is:

DEFINE (DEF) _____name-type;
(URL names)

Where the name-type may be one of the following:

ATTRIBUTE (ATTR) - defines a characteristic or mode of the target system.

ATTRIBUTE-VALUE (ATTV) - defines a particular value for an

associated ATTRIBUTE.

CLASSIFICATION (CLS) - can be associated with data processes and processors.

KEYWORD (KEY) - can be related to names for retrieval and analysis purposes.

MAILBOX (BOX) - defines an address for a PROBLEM-DEFINER.

SECURITY (SEC) - defines security status for one or more DPL names.

SOURCE (SRC) - defines a reference for additional information related to objects being described.

SUBSETTING-CRITERION (SSCN) - defines some data objects whose value is used as the criterion for segmenting a SET of data.

SYSTEM-PARAMETER (SYSP) - defines an object whose value influences the size of particular aspects of the system.

TRACE-KEY (TKEY) - can be used to relate names which exist in different data bases.

3.20.1 System-Structure Statements for the DEFINE Section

SUBSETTING-CRITERION names may be defined to apply to one or more SETS via the SUBSETTING-CRITERION statement.

SUBSETTING-CRITERION (SSCN) _____:
(list of set names)

No other name types in the DEFINE section may use this statement.

3.20.2 Data-Derivation Statements for the DEFINE Section

The MAINTAINED statement specifies those PROCESSES that maintain SUBSETTING-CRITERION for organization of a SET.

MAINTAINED (MNTD) _____:
(list of process names)

The MAINTAINED statement allows the use of the optional clauses DEPENDING ON and FOR EACH.

Maintenance of SUBSETTING-CRITERION involves placement of ENTITIES, INPUTS and OUTPUTS in proper SETS according to the values of the SUBSETTING-CRITERION contained within them.

No other name types in the DEFINE section may use this statement.

3.20.3 System-Size Statements for the DEFINE Section

SYSTEM-PARAMETERS and ATTRIBUTE-VALUES may be defined to have a VALUE or range of VALUES associated with them.

The VALUE statement is used to define the numeric values a SYSTEM-PARAMETER may have.

```
VALUE (VAL)----------------------------------------;
                  (integer value)
```

Only positive integer values may be specified. Decimal numbers, negative numbers, etc., are not acceptable.

A range of VALUES may also be specified.

```
VALUES (VAL) _____________THRU----------------;
          (minimum value)     (maximum value)
```

Again, the VALUES must be positive integers. The minimum value must be less than the maximum value. POSINF and NEGINF may be used to represent positive and negative infinity, respectively.

Only one VALUE statement, of either of the forms, may be given to describe a particular SYSTEM-PARAMETER.

No other name types in the DEFINE section may use this statement.

3.20.4 Project-Management Statements for the DEFINE Section

The RESPONSIBLE-PROBLEM-DEFINER statement may be used in this Section. Description and syntax for these statements are given in section 3.2.

3.20.5 System-Properties Statements for the DEFINE Section

The SYNONYMS, DESCRIPTION, SEE-MEMO, KEYWORDS, ATTRIBUTES, ASSERT, SECURITY, SOURCE and TRACE-KEY statements may be used in this Section. Description and syntax of these statements are given in section 3.2.

The APPLIES statement may be used for MAILBOXES, SECURITIES and SOURCES to specify the URL names that they apply to.

APPLIES (APP) _____;
(list of names)

Exceptions are that SECURITY may not have SECURITY, and SOURCES may not have a SOURCE.

3.21 The DESIGNATE Section

The DESIGNATE section consists of one statement which specifies that a given name is to be made a SYNONYM of another name.

This facilitates the advantage of using short abbreviations when referencing a particular object.

DESIGNATE (DESG) _____AS A SYNONYM (SYN);
(a name)

A name can have any number of SYNONYMS, but a name can be a SYNONYM for only one other name.

No other statements are allowed in this section.

4. STRATEGY IN USING URL

URL is a very flexible and comprehensive language. Most situations can be represented or expressed in URL in more than one way; each of which is syntactically correct. However, the different representatives may imply different semantics which may or may not be what the analyst intended. This section describes a number of situations in which alternative methods of expression are possible and outlines the implications of different strategies.

4.1 Specifying the "System" Boundary

In URL, a URA data base contains the description of one system. Each system has a boundary and the system description may be thought of as consisting of two parts:

- the specification of what goes on inside the system.

- the specification of what crosses the boundary.

Alternative strategies are possible in the order in which these parts are specified. One possibility is to delineate the boundary first. The second is to describe the "interior" of the system without identifying the boundary.

1) Specifying the Boundary First

A firm boundary is obtained when INTERFACES are defined and their communication with the system is specified by naming INPUTS and OUTPUTS. It is assumed here that INPUTS enter the system, and OUTPUTS leave the system, in some physical form containing data values. The constraint in URL is that an INPUT can only come from an INTERFACE and OUTPUTS only go to an INTERFACE. Inside the system, a number of PROCESSES may be named, each one of which uses data from the available sources - INPUTS, ENTITIES or SETS, or from unspecified sources - GROUPS and ELEMENTS, to derive and update data. A PROCESS may USE data from any of these sources or DERIVED from any PROCESS; and similarly, data DERIVED by one PROCESS may be USED by any number of other PROCESSES.

One benefit of this approach is that the problem statement can be checked for completeness, e.g., that each INPUT is GENERATED by some INTERFACE and RECEIVED by at least one PROCESS and that each OUTPUT is GENERATED by some PROCESS and RECEIVED by at least one INTERFACE. Another benefit is that the description of the INPUTS and OUTPUTS can be agreed to by the relevant INTERFACE.

A disadvantage of this approach is that an INTERFACE is not a

PROCESS and an object that is an INPUT to the system cannot also be an OUTPUT.

2) Specifying the Interior of the System First

In some cases, it may be desirable to delay specifying the system boundary until the interior of the system has been described. This can be done by not identifying any INPUTS and OUTPUTS, but instead, defining those PROCESSES that USE and DERIVE data and defining data in terms of ENTITIES, SETS, GROUPS and ELEMENTS. What would be an INTERFACE in the previous case, now can be identified as a PROCESS, and therefore, the object in the real world can use data from any source-derived data which can be used by any other PROCESS.

The advantage of this approach is that any of the objects, e.g., PROCESS, can both USE data and DERIVE data and that a given collection of data identified as an ENTITY can be both USED by a PROCESS and be DERIVED by a PROCESS (in addition of course to being UPDATED).

4.2 Assignment of Name Types.

URL requires that each name (object) used in the system description be of a certain type. There are 29 types available of which 20 are defined by their own sections and the other 9 are defined by the DEFINE section.

The assignment of a type to a name is crucial. Statements that can be made about an object and its relationship to other objects are limited to those available in the object's section. In some situations the choice of a type for a particular object is clear; in other situations there may be several legitimate choices. This section discusses the situation in which there are alternatives.

4.2.1 INTERFACES Versus PROCESSES

In very general terms, a PROCESS is an object which is part of the target system, and which operates on data values which it USES to DERIVE new data values. The PROCESS can also UPDATE data. The data which is used by a PROCESS can come "from" any other PROCESS, and the data which is DERIVED can be USED by any other PROCESS.

In contrast, an INTERFACE is a unit outside the boundary of the target system which can produce data for the target system (GENERATE an INPUT) and/or receive data from the target system (RECEIVE an OUTPUT).

An object, therefore, should be assigned an INTERFACE type name

only if it interacts with the target system, namely, that it will either RECEIVE or GENERATE data. Otherwise, the object should be assigned the name type "PROCESS."

4.2.2 INPUTS, OUTPUTS and ENTITIES

INPUTS, OUTPUTS and ENTITIES are types of objects which "contain" or "carry" sets or collections of data values. Conceptually, the name can represent both the "container" or the collection of data values contained in that container. Furthermore, the container can be regarded as physical, that is, a card, a tape, a record on a disc, etc., or it can be regarded as a logical construct which may or may not be physically implemented in that form.

An object should be designated as in INPUT if what is to be specified is a container with data values coming into the target system from outside, i.e., from an INTERFACE.

Another distinguishing characteristic of INPUTS and OUTPUTS is that when interpreted as "containers" of data values they are temporary as far as the target system is concerned. There may be multiple instances of the particular INPUT coming in, but once it is received by a PROCESS, the particular instance disappears.

For example:

INPUT time-card

implies that the system will receive objects of type INPUT which are called time-card. The number of individual 'time-cards' which arrive is specified by the statements such as HAPPENS.

Similarly, an object should be designated an OUTPUT if it is a "container" of data values and if it is specified to leave the target system. Again, there may be multiple instances, each one of which has to be GENERATED and each leaves the system. Once individual instances of the OUTPUT have left the target system, they are not considered part of, or accessible to, the target system.

The reasons for distinguishing INPUTS and OUTPUTS from ENTITIES and GROUPS is that (1) eventually the physical medium on which they appear and their representation will have to be specified, and (2) the source and destination can be related to INTERFACES, and (3) time and volume can be specified for INPUTS and OUTPUTS but not for GROUPS.

An ENTITY is a "container" of data value; in this respect it is equivalent to an INPUT or OUTPUT. However, it differs from INPUTS and OUTPUTS in that it is internal to the system and it persists. Therefore, an individual instance of an ENTITY must

be created, i.e., DERIVED.

Again, the ENTITY may be a "logical" collection of data values or it may be a "physical" collection. When it is designated as a physical collection, it will probably be implemented as a logical record or physical record which is maintained by the system in some way.

Therefore, an object which is a collection of data values that is internal to the target system and persists in the system, should be designated an ENTITY rather than as an INPUT or OUTPUT.

4.2.3 ENTITIES Versus GROUPS

An ENTITY is a logical collection of data values. Data values are particular instances of ELEMENTS or GROUPS. The data values included in the collection are specified by the CONSISTS of statement. A GROUP is also specified as CONSISTING of a number of GROUPS and/or data ELEMENTS.

The major distinction between ENTITIES and GROUPS lies in that ENTITY is a container of values of the ELEMENTS of which it CONSISTS. A GROUP, on the other hand, is merely a notational convenience for naming a set of data of which it CONSISTS. Whenever the analyst finds that a number of data ELEMENTS appear in a number of situations together, he can simplify his writing time and analysis time by defining the collection as a GROUP.

Other differences between ENTITIES and GROUPS are the following.

1) GROUPS can be CONTAINED in ENTITIES, INPUTS and OUTPUTS, but ENTITIES cannot.

2) ENTITIES (and INPUTS and OUTPUTS) can be CONTAINED in SETS, but GROUPS cannot.

3) ENTITIES can CONSIST of GROUPS but not of other ENTITIES. GROUPS can CONSIST of other GROUPS, but, of course, not of ENTITIES.

4) GROUPS can be used as SUBSETTING-CRITERIA of SETS and to IDENTIFY ENTITIES, but ENTITIES cannot. ENTITIES can be RELATED via RELATION statements and have ASSOCIATED data consisting of GROUPS.

5) As far as PROCESSES are concerned, the same statements that can be made about ENTITIES can also be made about GROUPS, though when the ENTITY is used in a statement, the appropriate statement about the ELEMENTS or GROUPS CONTAINED in the ENTITY must also be made (See Table 4.1).

6) Both ENTITIES and GROUPS can have SYSTEM-PARAMETERS

associated with the CONSISTS statement. In addition, the ENTITY can have a CARDINALITY and VOLATILITY statement while a GROUP cannot.

The problem definer should specify an object to be an ENTITY when he wishes to refer a number of ELEMENTS as a unit.

4.3 Selection of Relationships

All relationships in URL are precisely defined by statements. In many cases, only one statement will be legitimate. In some cases, however, there may be a choice. These situations are outlined in this section.

4.3.1 RECEIVES/GENERATES Versus USES/UPDATES/DERIVES

1) RECEIVES/GENERATES can only refer to INPUTS/OUTPUTS whereas, USES/UPDATES/DERIVES can only refer to "data."

2) USES implies that the data value of what is being USED must be available.

   UPDATES implies that the data value must be CONTAINED in an ENTITY. DERIVES implies a value is computed.

3) When INPUTS are USED, OUTPUTS are DERIVED,

   SETS are USED, UPDATED or DERIVED,

   ENTITIES are USED, UPDATED or DERIVED

   this implies that the data values in these "containers" of data values are being referred to.

4) When GROUPS are USED, UPDATED or DERIVED at least one element in the GROUP is referred to.

5) The allowable syntax for which statements can affect which objects is shown in Table 2.4.3.1. The meaning of the statements is shown in Table 2.4.3.2.

5. ACHIEVING GOOD DOCUMENTATION

Documentation of the target system, of its interfaces with the organization and its environment, and of the system development project is used for different purposes. Figure 5.0 outlines some characteristics of present manual documentation and some desirable characteristics that are achievable with computer-aided documentation.

To achieve the potential benefits of computer-aided documentation requires:

- a formal language which permits relationships to be precisely defined.

- a computer program which provides a method for enforcing correct use of the formal language.

- good procedures to be followed by the analyst.

The last of these is important since no matter how good the language and the computer software, the benefits will never be attained unless the tools are used properly.

In Section 5.1 the characteristics of good documentation are described and methods are suggested by which the analyst can achieve them using URL/URA. Section 5.2 summarizes the checks for preciseness and consistency which are performed by the Analyzer. Checks which the analyst can perform using the outputs available from the Analyzer are described in Section 5.3.

5.1 Characteristics of Good Documentation

Usually, the analyst is responsible for producing documentation. This section outlines some major attributes of good documentation and indicates how an analyst may use URL/URA to achieve them.

5.1.1 Understandability

Documentation with this characteristic is in an easy-to-read format and is presented at a general enough level so that persons, no matter what their background, should be able to read and comprehend the material within.

Reports can be generated from the problem statement in several common formats, e.g., flow diagrams, matrices and at different levels of detail. For example, it is often desirable to initially present high level objects and have subsequent reports present more and more detail about these objects until

everything is described in terms of their lowest level constituents. The analyst can choose the ordering and content of the reports.

Present Manual Documentation	Desirable Characteristics of Computer-Aided Documentation
Hard to Understand	Understandable
Ambiguous	Precise
Inconsistent	Consistent
Incomplete	Complete
Incorrect	Correct
Difficult to Analyze and Evaluate	Computer-Aided Analysis and Evaluation
Hard to Modify	Computer-Aided Updating

Figure 5.0 Characteristics of Documentation

5.1.2 Preciseness

Documentation with this quality must have all relevant terminology explicitly defined so that information presented cannot be misinterpreted.

A computer interpreted language must have an accurately defined syntax. The reserved words in the syntax of URL are used to describe different objects and the relationships between the objects. Definitions of all reserved words allowed in the syntax are fixed so that all relationships presented in the documentation (URA reports) are exactly the same as those initially specified by the analyst (i.e., there can only be one interpretation of the information).

5.1.3 Consistency

Documentation which is "consistent" presents all the material in proper context and does not have statements that are conflicting, contradictory or inconsistent.

The context in which a particular object is to be used is defined by the user via URL statements which will be stated in the URA data base. Any attempts to use the previously defined object in a conflicting context will result in an error diagnostic. Therefore, use of URA maintains consistency throughout the documentation.

5.1.4 Completeness

To be "complete," documentation must present the material in sufficient detail so that no reference to outside sources is needed for a thorough understanding of the subject matter. Every necessary piece of information must be available and no relationship must be left dangling.

URL allows a number of relationships and objects to be defined to describe an Information Processing System. The URL statements offered provide a thorough outline of what should be incorporated into the documentation of an IPS. The statements in URL facilitate the enforcement of completeness.

5.1.5 Correctness

To be "correct," the analyst must insure that all relationships specified in the documentation are valid, and that all information recorded is true.

The syntax rules enforced by URA insures that all relationships in the documentation are valid. Though it is impossible to know whether the information recorded is true or not, many of the reports available can present the information in a format easy for the analyst to check for errors (e.g., misspellings, incorrect narrative descriptions, etc.).

5.1.6 Analyzability

Documentation which is analyzable must be organized in such a way that any information not explicitly stated in it must be easily derived through some procedure.

Since all URL statements are stored in a data base, all data is easily accessed and can be presented in the form of a URA report. In addition, any new developments in analyzing the information (e.g., Cost/Benefit Analysis, etc.) can be incorporated into the existing URA package.

5.1.7 Ease of Modification

Documentation which is easy to modify must have sufficient indexing facilities so that all occurrences of a given item in the documentation may be referenced if and when a change to the item is required.

Because the information used in deriving URA reports is contained within the URA data base, any modifications to the data base will be reflected in reports produced after the change. URA offers several commands to modify the data base. Any reports generated after the modifications will be

up-to-date.

5.2 Checks Carried Out by the Analyzer

For the most part, the characteristics of good documentation can be realized when the documentation is generated by computer-aided means. Preciseness, consistency, and correctness are all checked by the Analyzer as new information is added to the data base or data is modified in it.

URA can produce several hundred diagnostic and error messages. Each is identified by a number. The complete list is given in the "User Requirements Analyzer User's Manual" in numerical order to facilitate correction. Here the error messages are analyzed in terms of how they contribute to good documentation.

5.2.1 Checks Related to Preciseness

A considerable portion of the error detection facilities in the Analyzer are used to check the "preciseness" of new URL statements being added to the data base. (This is done each time RP-URL is initiated.) The Analyzer must check that the syntax is correct and that the user-defined names given in the new statements are consistent with names already in the data base. If either of these conditions fail, an error diagnostic must be generated by the Analyzer to inform the user that the information to be stored in the data base was ambiguous or inconsistent with the information already in the data base. No ambiguous or inconsistent information is stored in the data base.

a) Syntax Errors

Breaking any of the syntax rules of URL will cause the Analyzer to generate one or more error diagnostics. Typical syntax errors are:

- use of illegal characters.

- misspelling of URA reserved words.

- omission of semi-colon to terminate line.

Table 5.2.1 presents a complete list of diagnostics produced when a syntax error is encountered.

b) Incorrect Use of Names

It is very important that once a name is defined and has an associated name type along with it (e.g., PROCESS or SET), the name can only be used in the context in which it was

defined. Therefore, a name defined to be a PROCESS cannot also be used to define a GROUP of data. Likewise, only those relationships specified by the "User Requirements Language, Language Reference Manual" can be used to relate to objects. For example, a USES relationship between two PROCESS names is not allowed and any attempt to specify this would cause the Analyzer to generate the diagnostic:

MUST BE ELEMENT, GROUP, INPUT, ENTITY OR SET

Table 5.2.1.1 presents a complete list of the errors that can be encountered when incorrectly using names.

5.2.2 Checks Associated with Consistency

As URL statements are being added to the data base, the Analyzer also checks that the new relationships being specified are consistent with the information already in the data base. In the previous section, the Analyzer was shown to check that once a name was defined to be a given name type, it could not be used in a conflicting context (i.e., as a different name type). The Analyzer must also check that the relationships specified for a given name do not conflict. For example, if an ENTITY was defined to have a CARDINALITY of 100, it would be illogical to also say that its CARDINALITY is 50. The Analyzer will detect these types of inconsistencies. The Consistency Error Messages are listed in Table 5.2.2. Table 5.2.2.1 presents the various inconsistencies detected by the Analyzer according to name type and relationships within the system description.

Error Number		Diagnostic
2	NLEX	NAME TOO LONG
3	NLEX	'EOF' NOT FOUND BEFORE END-OF-FILE
4	INDBS	ERROR OPENING DATA BASE FOR -
5	NLEX	END-OF-FILE IN MIDDLE OF COMMENT
7	SCAN	ILLEGAL CHARACTER - IGNORED
10	PRDUCT	NO APPLICABLE PRODUCTION - SYNTAX ERROR - START SKIPPING
11	STACK	ILLEGAL SYMBOL PAIR - SYNTAX ERROR - START SKIPPING
16	COMENT	END-OF-FILE IN COMMENT ENTRY
90	RWLIST	SSCN IS ONLY LEGAL TYPE IN DEFINE SECTION WHICH CAN BE MAINTAINED
114	VLIST	ONLY SINGLE VALUE OR RANGE ALLOWED - IGNORED
116	OTHERS	VALUES ONLY LEGAL FOR ELEMENT, SYSPAR, OR ATTRIBUTE-VALUE
119	CLSCA	PUNCH= NOT ALLOWED IN THIS IMPLEMENTATION
221	PLIST	NAME NOT PART OF HEADER
225	RWLIST	CANNOT HAVE KEYWORD FOR KEYWORD
228	RWLIST	CANNOT HAVE SECURITY FOR SECURITY
229	RWLIST	CANNOT HAVE SOURCE FOR SOURCE
231	RWLIST	SYNONYMS ONLY APPLIED TO FIRST NAME
232	APPLES	APPLIES STATEMENT ILLEGAL WITH THIS NAME TYPE
266	ILLST	ILLEGAL STATEMENT IN THIS SECTION

Table 5.2.1
URL Syntax Error Messages

Error Number		Diagnostic
25	HEAD	INVALID HEADER STATEMENT - STATEMENTS WILL BE IGNORED
51	RWLIST	MUST BE SUBSETTING-CRITERION NAME
101	NLIST2	NAME ALREADY USED IN DIFFERENT CONTEXT
102	NLIST2	NAME ALREADY USED IN DIFFERENT CONTEXT
118	CLRCA	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
202	NLIST	NAME PREVIOUSLY USED DIFFERENTLY - IGNORED
204	DFFN	NAME ALREADY USED IN DIFFERENT CONTEXT
207	SETSYN	CANNOT BE MADE SYNONYM - DIFFERENT TYPES
209	CHKCON	STACK OVERFLOW WHILE WALKING CONSISTS STRUCTURE
210	PRTNUN	NO NAMES IN DATA BASE
211	OTHERS	NAME MUST BE ENTITY NAME
216	OTHERS	NAME MUST BE ENTITY NAME BEFORE VIA
217	OTHERS	NAME MUST BE RELATION AFTER VIA
219	CLSRCA	FILE= NOT ALLOWED IN THIS IMPLEMENTATION
234	OPTRW	NAME ALREADY USED IN DIFFERENT CONTEXT
235	OPTION	NAME ALREADY USED IN DIFFERENT CONTEXT
236	OPTION	NAME LIST TOO LONG - REST IGNORED
240	APPLES	KEYWORD CANNOT APPLY TO KEYWORD
241	APPLES	MAILBOX CAN ONLY APPLY TO PD
246	APPLES	SECURITY CANNOT APPLY TO SECURITY
247	APPLES	SOURCE CANNOT APPLY TO SOURCE
248	APPLES	MEMO CANNOT APPLY TO MEMO
257	FORMSL	NAME NOT IN DATA BASE -
267	ILLST	NO CURRENT SECTION

Table 5.2.1.1
UPL Name Error Messages

Error Number		Diagnostic
22	RWLIST	SAME ATTRIBUTE ALREADY GIVEN WITH DIFFERENT ATTRIBUTE VALUE
43	OTHERS	CARDINALITY ALREADY GIVEN AS SYSPAR
44	OTHERS	CARDINALITY ALREADY GIVEN AS DIFFERENT VALUE
60	APPLES	SECOND MAILBOX FOR RD ILLEGAL
61	RWLIST	ALREADY PART OF SOMETHING ELSE
62	RWLIST	SECOND RD FOR THIS NAME ILLEGAL
63	RWLIST	ALREADY PART OF SOMETHING ELSE
115	VLIST	MIN NOT LESS THAN MAX - IGNORED
117	OTHERS	DIFFERENT VALUES ALREADY GIVEN
205	SETSYN	ALREADY SYNONYM FOR SOMETHING ELSE
212	OTHERS	RELATION ALREADY EXISTS BETWEEN TWO OTHER ENTITIES
213	OTHERS	CAN HAVE ONLY ONE CARDINALITY
214	OTHERS	CONNECTIVITY ALREADY GIVEN FOR THIS RELATION
215	RWLIST	ALREADY CONTAINS WITH DIFFERENT SYSTEM PARAMETER
218	OTHERS	RELATION ALREADY EXISTS BETWEEN DIFFERENT ENTITY
265	RWLIST	CONNECTION ALREADY EXIST WITH DIFFERENT VALUE OR NAME

Table 5.2.2
UBI Consistency Error Messages

	System Flow	System Structure	Data Structure	Data Derivation
RWE		61,63		
INPUT		61,63		
OUTPUT		61,63		
ENTITY			212,218	
RELATION			212,218	
PROCESS		61,63		
OTHER	205	205	205	90,205

TABLE 5.2.2.1
CONSISTENCY ERRORS

	System Size	System Dynamics	System Properties	Project Management
RWR				62
INPUT	215			62
OUTPUT	215			62
SET	43,44 213,215			62
ENTITY	43,44 213,214			62
RELATION	43,44 213,214			62
GROUP	215			62
ELEMENT	117,115			62
PROCESS				62
INTERVAL	215			62
SYSTEM PARAMETER	265,115			62
EVENT				62
CONDITION				62
CONDITION				62
OTHER	205	205	22,205	60,62,205

TABLE 5.2.2.1 (continued)

5.3 Consistency and Completeness Checks Carried Out by the Analyst

At some point in time in the development of the problem statement, the Analyst may want to check its state of consistency and/or completeness. The Analyst can perform these checks by inspection of various reports available from the Analyzer. This technique is possible because all information specified in the data base can be presented via one or more reports. Since the Analyzer has checked all inputs to the data base for syntax and consistency errors, the problem statement presented is always in a correct state. It is the role of the Analyst to determine whether it is totally "consistent" or "complete."

Table 3.5 presents a summary of all consistency and completeness checks to be carried out by the Analyst.

Table 5.3.1 presents a summary of the benefits of particular URA reports in identifying inconsistencies and incompleteness in the problem statement.

I) SYSTEM FLOW
   a) All INTERFACES should GENERATE some INPUT, RECEIVE some OUTPUT, or be RESPONSIBLE for some SET.
   b) All INPUTS should be GENERATED by at least one INTERFACE.
   c) All INPUTS should be RECEIVED by at least one PROCESS.
   d) All OUTPUTS should be GENERATED by at least one PROCESS.
   e) All OUTPUTS should be RECEIVED by at least one INTERFACE.

II) SYSTEM STRUCTURE
   a) All PROCESSES without SUBPARTS should have PROCEDURES.
   b) SETS with SUBSETS should have SUBSETTING-CRITERIA.
   c) All INPUTS without SUBPARTS should be broken down via the CONSISTS statement.
   d) All OUTPUTS without SUBPARTS should be broken down via the CONSISTS statement.

III) DATA STRUCTURE
   a) All ELEMENTS should be available from an INPUT or from a ENTITY, or DERIVED by some PROCESS.
   b) All SETS should CONSIST of INPUTS, OUTPUTS or ENTITIES.
   c) All ENTITIES should be broken down via the CONSISTS statement.
   d) All INPUTS should be broken down via the CONSISTS statement if there are no SUBPARTS.
   e) All OUTPUTS should be broken down via the CONSISTS statement if there are no SUBPARTS.
   f) All RELATIONS should have a BETWEEN statement.
   g) All GROUPS should be composed of ELEMENTS.

Table 5.3a
Summary of Completeness Checks to be made by Analyst

IV) DATA DERIVATION
- a) All ELEMENTS should be USED, UPDATED and/or DERIVED by at least one PROCESS.
- b) All PROCESSES should acquire information by RECEIVING, USING or UPDATING.
- c) All PROCESSES should produce information by GENERATING, DERIVING, or UPDATING.
- d) All SETS should be USED, UPDATED or DERIVED by at least one PROCESS.
- e) All SETS should have a DERIVATION statement.
- f) All ENTITIES should be USED, UPDATED or DERIVED.
- g) All ELEMENTS within an INPUT should be USED.
- h) All ELEMENTS within an OUTPUT should be DERIVED.
- i) All ELEMENTS within an ENTITY should be USED, UPDATED or DERIVED.
- j) All RELATIONS should be MAINTAINED by at least one PROCESS.
- k) All RELATIONS should have a DERIVATION statement.

V) SYSTEM SIZE AND VOLUME
- a) All EVENTS should have a HAPPENS statement.
- b) All PROCESSES should have a HAPPENS statement.
- c) All SETS should have a CARDINALITY statement.
- d) All SETS should have a VOLATILITY-SET statement.
- e) All SETS should have a VOLATILITY-MEMBER statement.
- f) All ENTITIES should have a CARDINALITY statement.
- g) All ENTITIES should have a HAPPENS statement.
- h) All INPUTS should have a HAPPENS statement.
- i) All OUTPUTS should have a HAPPENS statement.
- j) All RELATIONS should have a CARDINALITY statement.
- k) All RELATIONS should have a CONNECTIVITY statement.

VI) SYSTEM DYNAMICS
- a) Each EVENT should be associated with at least one CONDITION or PROCESS.
- b) Each CONDITION should be associated with at least one EVENT or PROCESS.
- c) Each CONDITION should have a TRUE WHILE or FALSE WHILE statement.

VII) SYSTEM PROPERTIES
- a) All KEYWORDS, ATTRIBUTES, SOURCES, SECURITIES and TRACE-KEYS should APPLY to some other URL names.

VIII) PROJECT MANAGEMENT
- a) All PROBLEM-DEFINERS should have a MAILBOX.
- b) All PROBLEM-DEFINERS should be RESPONSIBLE for the description of at least one URL objects.

Table 5.3a (continued)

Analyzer Commands	Completeness Checks
ATTRIBUTE INFORMATION REPORT	VIIa
CONSISTS COMPARISON MATRIX	IIIc, IIId, IIIe, IIIg
CONTENTS REPORT	IIIc, IIId, IIIe, IIIg
DATA PROCESS REPORT	Ic, Id; IVa, IVb, IVc, IVd, IVf
FORMATTED PROBLEM STATEMENT	Ia-Ie; IIa-IId; IIIa-IIIg; IVa-IVf, IVi, IVj; Va-Vk; VIa-VIc; VIIIb
FREQUENCY REPORT	Va, Vb, Vh, Vi
NAME GEN	VIIIa, VIIIb
PICTURE	Ia, Ib, Ic, Id, Ie; IIb, IIC, IId
PROCESS INPUT/OUTPUT	IVb, IVc
PUNCHED COMMENT ENTRIES	IVe, IVj, Vd, Ve, Vg

Table 5.3h
URA Reports that may be used by Visually Check
for Completeness of the Problem Statement

URA Report	Completeness Checks
CONSISTS COMPARISON MATRIX	- All INPUTS, OUTPUTS, ENTITIES and GROUPS are broken down to ELEMENTS at the lowest level. - All necessary ELEMENTS are defined in the data structure for a particular INPUT, OUTPUT or ENTITY.
CONSISTS MATRIX	- All GROUPS and ELEMENTS belong to higher level data structures. - All SETS broken into INPUTS, or OUTPUTS or ENTITIES[1]
CONTENTS REPORT	- All INPUTS, OUTPUTS, ENTITIES and GROUPS are broken down to ELEMENTS at the lowest level. - All SETS broken into INPUTS, or OUTPUTS ENTITIES
DATA PROCESS REPORT	- All INPUTS RECEIVED by some PROCESS[1] - All INPUTS USED by some PROCESS[1] - All OUTPUTS GENERATED by some PROCESS[1] - All OUTPUTS DERIVED by some PROCESS[1] - All ENTITIES and SETS DERIVED by some PROCESS[1] - All ENTITIES and SETS DERIVED and USED by some PROCESS[1] - All ENTITIES and SETS are UPDATED and USED by some PROCESS[1] - All GROUPS and ELEMENTS are DERIVED or UPDATED or USED by some PROCESS[1] - All PROCESSES USE data and DERIVE or UPDATE data[1] - All PROCESSES which DERIVE data also USE data[1] - All PROCESSES which UPDATE data also USE data[1] - All PROCESSES interact with data in some way[1]
DICTIONARY REPORT	- All names should have a narrative DESCRIPTION and RESPONSIBLE-PROBLEM-DEFINER
DYNAMIC ANALYSIS	- All the dynamic relations for CONDITIONS, EVENTS, PROCESS and INPUTS are broken down to the lowest level.

Table 5.3.1
Completeness and Consistency Checks Made by URA Reports

---

[1] Computer-aided analysis

EXTENDED PICTURE	- All SETS are broken into ENTITIES or INPUTS or SETS - All PROCESS interact with data in some manner - All INTERFACES generate INPUTS to the SYSTEM and/or receive OUTPUTS - All OUTPUTS are generated - All INPUTS generated must be used in some manner - All SETS are used, updated or derived - All INPUT, OUTPUT, ENTITY are produced and/or used in some manner - All GROUP, ELEMENT are produced and/or used in some manner - All INPUT, OUTPUT, GROUP, ENTITY are eventually broken down into elements - All GROUP, ELEMENT are contained within some larger data.
IDENTIFIER INFORMATION REPORT	- Determines which ENTITIES have and do not have IDENTIFIERS
INTERVAL CONSISTENCY	- All INTERVALS are broken down into INTERVALS at the lowest level
FORMATTED PROBLEM STATEMENT	- The description of each name can be checked against all possible statements for that name.
FREQUENCY REPORT	- All INPUTS, OUTPUTS, PROCESSES and EVENTS should have a HAPPENS statement
KWIC INDEX	
NAME GEN	- All names of a particular type (e.g., PROCESS) have been defined for a particular problem statement
NAME LIST	- Names which have synonyms in the real world should have them in the problem statement
PICTURE	[given in Table 2.1b] - All names should be involved in structure and/or information flow of the problem statement[1]

Table 5.3.1 (Continued)

---

[1] Computer-aided analysis

FREQUENCY REPORT	- Determines whether or not the manner in which frequencies (HAPPENS statement) are assigned is consistent.
KWIC INDEX	- Determines whether or not conventions used in assigning names is consistent
NAME GEN	- Determines whether or not naming is consistent - Determines whether or not name types have been assigned correctly.
NAME LIST	- Determines whether or not naming is consistent - Determines whether or not name types have been assigned correctly - Determines whether or not SYNONYMS have been assigned correctly
PICTURE	- Determines whether or not the name the PICTURE is generated for relates to the structure and information flow aspects of the problem statement correctly
ATTRIBUTE REPORT	- Determines whether or not the conventions of assigning ATTRIBUTES is consistent
PROCESS INPUT/ OUTPUT	- Determines whether or not the manner in which PROCESSES are described is consistent
PROCESS CHAIN REPORT	- Determines whether or not the name the PROCESS-CHAIN is generated for relates to the dynamic aspects of the problem statement correctly.

Table 5.3.1 (Continued)